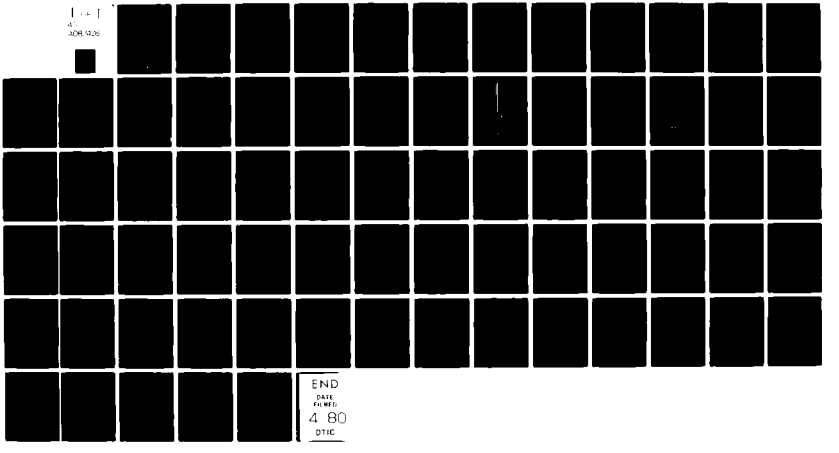
AD-A082 426 BOEING AEROSPACE CO SEATTLE WA F/G 9/2
MEASURING AND REPORTING SOFTWARE STATUS, ONE OF THE SOFTWARE AC--ETC(U)
AUG 78 R B DOGGETT, M P KRESS, K I MEHRER F33657-76-C-0723
UNCLASSIFIED D180-24728-1 ASD-TR-78-49 NL



END
DATE FILMED
4 80
DTIC

ASD-TR-78-49

MEASURING AND REPORTING SOFTWARE STATUS
One of the Software Acquisition Engineering Guidebook Series

DIRECTORATE OF EQUIPMENT ENGINEERING
DEPUTY FOR ENGINEERING

AUGUST 1978

TECHNICAL REPORT ASD-TR-78-49
Final Report

Approved for public release; distribution unlimited.

AERONAUTICAL SYSTEMS DIVISION
AIR FORCE SYSTEMS COMMAND
WRIGHT-PATTERSON AIR FORCE BASE, OHIO 45433

NOTICE

When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely related Government procurement operation, the United States Government thereby incurs no responsibility nor any obligation whatsoever; and the fact that the government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data, is not to be regarded by implication or otherwise as in any manner licensing the holder or any other person or corporation, or conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

This report has been reviewed by the Information Office (OI) and is releasable to the National Technical Information Service (NTIS). At NTIS, it will be available to the general public, including foreign nations.

This technical report has been reviewed and is approved for publication.

Richard W. Ittelson

RICHARD W. ITTELSON,
Technical Advisor
Directorate of Equipment Engineering

Richard J. Sylvester

RICHARD J. SYLVESTER,
ASD Weapon Systems Computer Resource
Focal Point
Deputy for Engineering

FOR THE COMMANDER

John S. Kubin

JOHN S. KUBIN, Colonel, USAF
Director, Equipment Engineering

"If your address has changed, if you wish to be removed from our mailing list, or if the addressee is no longer employed by your organization please notify _____,W-PAFB, OH 45433 to help us maintain a current mailing list".

Copies of this report should not be returned unless return is required by security considerations, contractual obligations, or notice on a specific document.

AIR FORCE/56780/17 March 1980 — 400

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER: ASD-TR-78-49	2. GOVT ACCESSION NO. / 3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle): MEASURING AND REPORTING SOFTWARE STATUS, One of the Software Acquisition Engineering Guidebook Series.	5. TYPE OF REPORT & PERIOD COVERED: Final rept.
	6. PERFORMING ORG. REPORT NUMBER: D180-24728-1
7. AUTHOR(s): R. B. Doggett, M. P. Kress, K. I. Mehrer	8. CONTRACT OR GRANT NUMBER(s): F33657-76-C-0723
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Boeing Aerospace Company, PO Box 3999, Seattle, Washington 98124	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: PE64740F, Project 2238
11. CONTROLLING OFFICE NAME AND ADDRESS: HQ ASD/ENE, Wright-Patterson AFB, Oh 45433	12. REPORT DATE: Aug[illegible]78 / 13. NUMBER OF PAGES: 80
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)	15. SECURITY CLASS. (of this report): UNCLASSIFIED / 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for Public Release, Distribution Unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Software Acquisition, Software Engineering, Software Scheduling, Software Audits, Software Management, Computer Program Configuration Items, Computer Program Development Cycle, Estimating Model, Life Cycle Analysis, Program Planning and Control, Unit Development Folder

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report is one of a series of guidebooks whose purpose is to assist Air Force Program Office Personnel and other USAF acquisition engineers in the acquisition engineering of software for Automatic Test Equipment and Training Simulators. This guidebook describes the status measuring and reporting functions. Emphasis is given to schedule control, technical performance monitoring and USAF audits.

DD FORM 1 JAN 73 1473

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

059610

FOREWORD

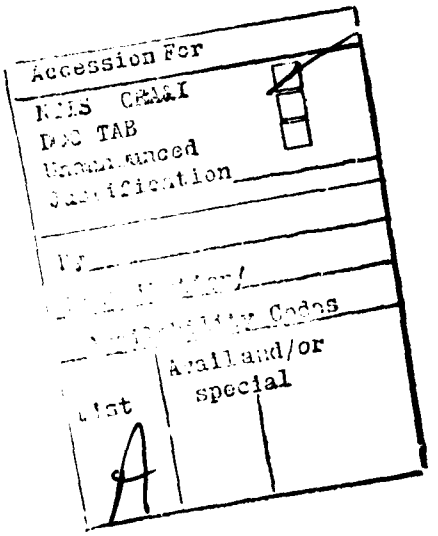
This guidebook was prepared as part of the Software Acquisition Engineering Guidebooks contract, F33657-76-C-0723. It describes the status measurement and reporting associated with Air Force/Contractor software procurement as applied to Training Simulators and Automatic Test Equipment. It is primarily intended for use by USAF acquisition engineering personnel.

This guidebook is one of a series intended to assist the Air Force Program Office and engineering personnel in software acquisition engineering for automatic test equipment and training simulators. Titles of other guidebooks in the series are listed in the introduction. These guidebooks will be revised periodically to reflect changes in software acquisition policies and feedback from users.

This guidebook reflects an interpretation of DOD directives, regulations and specifications which were current at the time of guidebook authorship. Since subsequent changes to the command media may invalidate such interpretations the reader should also consult applicable government documents representing authorized software acquisition engineering processes.

This guidebook contains alternative recommendations concerning methods for cost-effective software acquisition. The intent is that the reader determine the degree of applicability of any alternative based on specific requirements of the software acquisition with which he is concerned. Hence the guidebook should only be implemented as advisory rather than as mandatory or directive in nature.

This guidebook was prepared by the Boeing Aerospace Company.



This Software Acquisition Engineering Guidebook is one of a series prepared for Aeronautical Systems Division, Air Force Systems Command, Wright-Patterson AFB OH 45433. Inquiries regarding guidebook content should be sent to ASD/ENE, Wright-Patterson AFB OH 45433. The following list presents the technical report numbers and titles of the entire Software Acquisition Engineering Guidebook Series. Additional copies of this guidebook or any other in the series may be ordered from the Defense Documentation Center, Cameron Station, Alexandria VA 22314.

ASD-TR-78-43,	Computer Program Maintenance
ASD-TR-78-44,	Software Cost Measuring and Reporting
ASD-TR-78-45,	Requirements Specification
ASD-TR-78-46,	Computer Program Documentation Requirements
ASD-TR-78-47,	Software Quality Assurance
ASD-TR-78-48,	Software Configuration Management
ASD-TR-78-49,	Measuring and Reporting Software Status
ASD-TR-78-50,	Contracting for Software Acquisition
ASD-TR-79-5042,	Statements of Work (SOW) and Requests for Proposal (RFP)
ASD-TR-79-5043,	Reviews and Audits
ASD-TR-79-5044,	Verification, Validation and Certification
ASD-TR-79-5045,	Microprocessors and Firmware
ASD-TR-79-5046,	Software Development and Maintenance Facilities
ASD-TR-79-5047,	Software Systems Engineering
ASD-TR-79-5048,	Software Engineering (SAE) Guidebooks Application and Use

TABLE OF CONTENTS

TABLE OF CONTENTS - Continued

LIST OF FIGURES

Section 1.0 INTRODUCTION

Numerous DOD studies and publications have focused on the importance of software management visibility early in project development stages. Although projects are staffed with competent personnel, planning and organization may be lacking such that problems can develop, spread and seriously impact a project. Techniques for early problem recognition, planned abatement methods and follow-up are necessary to resolve both predictable and unforeseen software development difficulties. This guide book presents methods applicable to Automatic Test Equipment (ATE) and Training Simulator (TS) system software acquisition for measuring and reporting software development problems and correcting them before they cause major system problems.

1.1 PURPOSE

The purpose of measuring and reporting is to discover and correct problems threatening the project. This guidebook identifies the parameters which measure software development progress and describes methods of reporting and presenting that progress. It further describes methods of recognizing and correcting software problems. It is an aid to USAF planners in imposing requirements for, and participating in, contractor management visibility activities.

1.2 SCOPE

This is one of a series of guidebooks related to the Software Acquisition Engineering (SAE) process for TS and ATE ground based systems. The SAE guidebook titles are listed below:

- Software Cost Measuring and Reporting
- Requirements Specification
- Contracting for Software Acquisition
- Statement of Work (SOW) and Requests for Proposal (RFP)
- Regulations, Specification and Standards
- Measuring and Reporting Software Status
- Computer Program Documentation Requirements
- Software Quality Assurance
- Verification
- Validation and Certification
- Computer Program Maintenance
- Software Configuration Management
- Reviews and Audits
- Management Reporting by the Software Director

This guidebook covers: AF and contractor management visibility planning; factors to conside in measuring software development progress; methods of reporting status; and recognizing, categorizing and correcting problems with emphasis on the unique aspects of ATE and TS software.

Status monitoring methods discussed in this guidebook cover activities beginning with the full scale development phase (i.e. analysis, design, code and checkout, test and integration, installation and operation and support) as described in the Computer Program Development Plan (CPDP) and as covered in the guidebook on "Software Cost Measuring and Reporting."

1.3 TS AND ATE OVERVIEW

This section provides a brief sketch of TS and ATE system characteristics, including the function of the software associated with each.

1.3.1 TS System Characteristics

The TS system is a combination of specialized hardware, computing equipment, and software designed to provide a synthetic flight and/or tactics environment in which aircrews learn, develop and improve the skills associated with individual and coordinated tasks in specific mission situations. Visual, aural, and/or motion systems may be included. Figure 1.3-1 depicts a typical TS which employs digital processing capability.

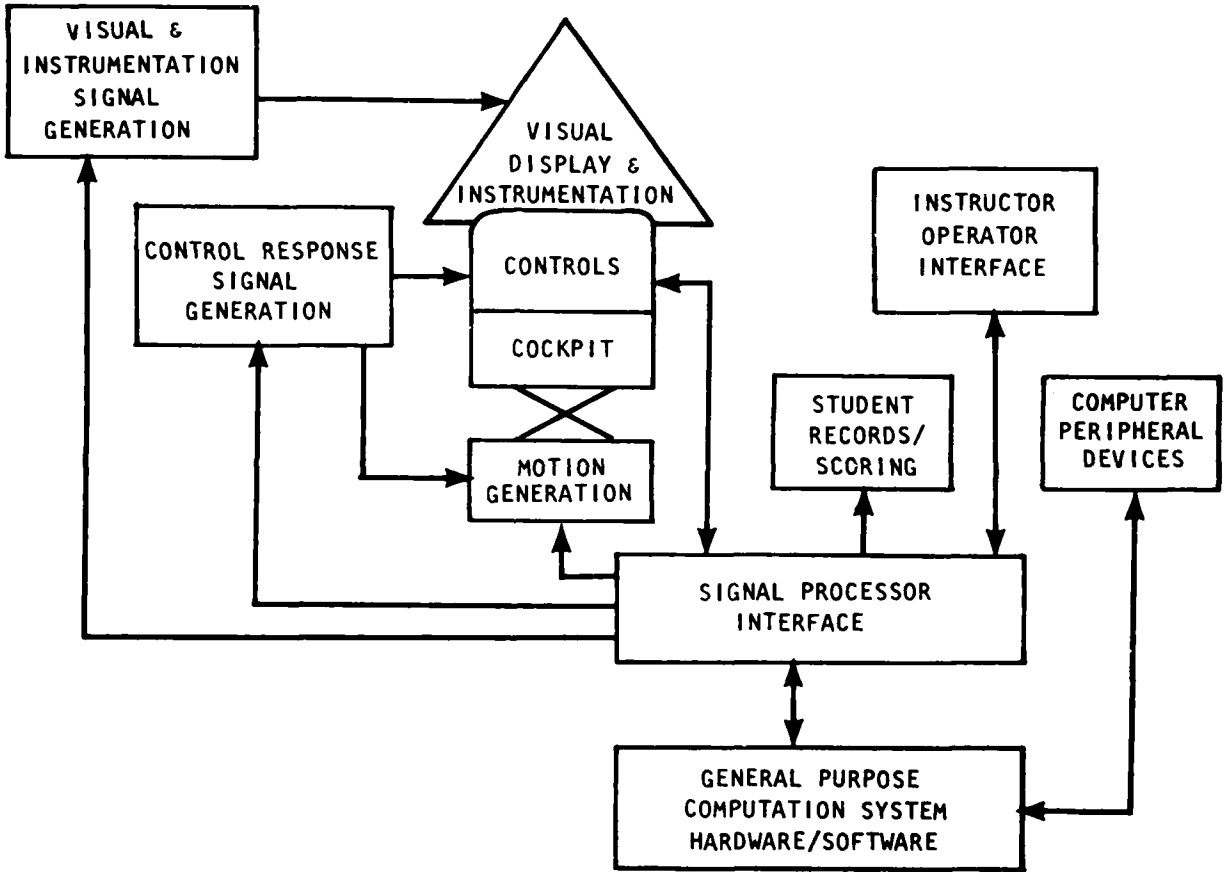


Figure 1.3-1. Typical Crew Training Simulator

The computer system, integral to the new TS can consist of one or more general purpose computers. The computing hardware operates with floating point arithmetic and sufficient bit capacity to provide efficient use of a simulator Higher Order Language (HOL).

When a multi-processor/multi-computer system is used, it must be designed such that computers can operate simultaneously and are controlled/synchronized by a single program (supervisor/executive). The executive directs program execution and regulates priorities. The simulator (1) accepts control inputs from the trainee (via crew station controls) or from the instructor operator station; (2) performs a realtime solution of the simulator mathematical model; and (3) provides output responses necessary to accurately represent the static and dynamic behavior of the real world system (within specified tolerance and performance criteria).

Since TS consist of interdependent hardware and software, a joint hardware/software development effort is required. As the complexity of TS increase, simulation software continues to grow in complexity, size, and cost. Software costs can and do exceed computer hardware costs in many cases. Therefore, it is imperative that the simulation software acquisition engineering process be subjected to formal system engineering planning and discipline to ensure cost-effective procurement.

1.3.2 ATE System Characteristics

ATE is defined as that ground support system which performs vigorous system tests with minimum manual intervention. ATE is used in place of manual devices because it is more cost effective, provides required repeatability, or repair of the item being tested requires the speed which only an automatic tester can achieve. Figure 1.3-2 shows the typical components of an ATE system. Note that there are both hardware and software elements involved. Most of the elements shown in the figure will be found in the majority of ATE systems although the packaging and interface design may vary between specific systems.

The controls and displays section consists of the computer peripheral devices such as control panels, magnetic tape cassettes or disks, a cathode ray tube (CRT), keyboard, and small printer. The computer (normally a minicomputer), as controlled by software, operates the peripheral devices; switches test stimuli on and off; and measures responses of the Unit Under Test (UUT) (comparing to predetermined values). The operator maintains supervisory control of the testing process through the peripherals. However, his interaction is usually minimal since, by definition, the automatic test feature was selected in preference to an operator-controlled test system.

ATE is normally designed to accomodate testing several different articles of system equipment (normally one at a time). The maintenance level being supported by the ATE is determined by logistics systems analysis. The importance of the software portion of the ATE system should not be minimized since both the application of the test stimuli and the measurement of the result are achieved via software. Arbitrary function generation and complicated wave analysis can also be accomplished by software and is becoming more prevalent in ATE systems. The cost of ATE software is a significant component of total ATE costs and design trades can be performed to minimize ATE life-cycle costs.

1.4 GUIDEBOOK ORGANIZATION

The guidebook is organized as follows. Section 1.0 is introductory. Section 2.0 identifies government and industry publications and data items dealing with management visibility for software development. Section 3.0 discusses planning for

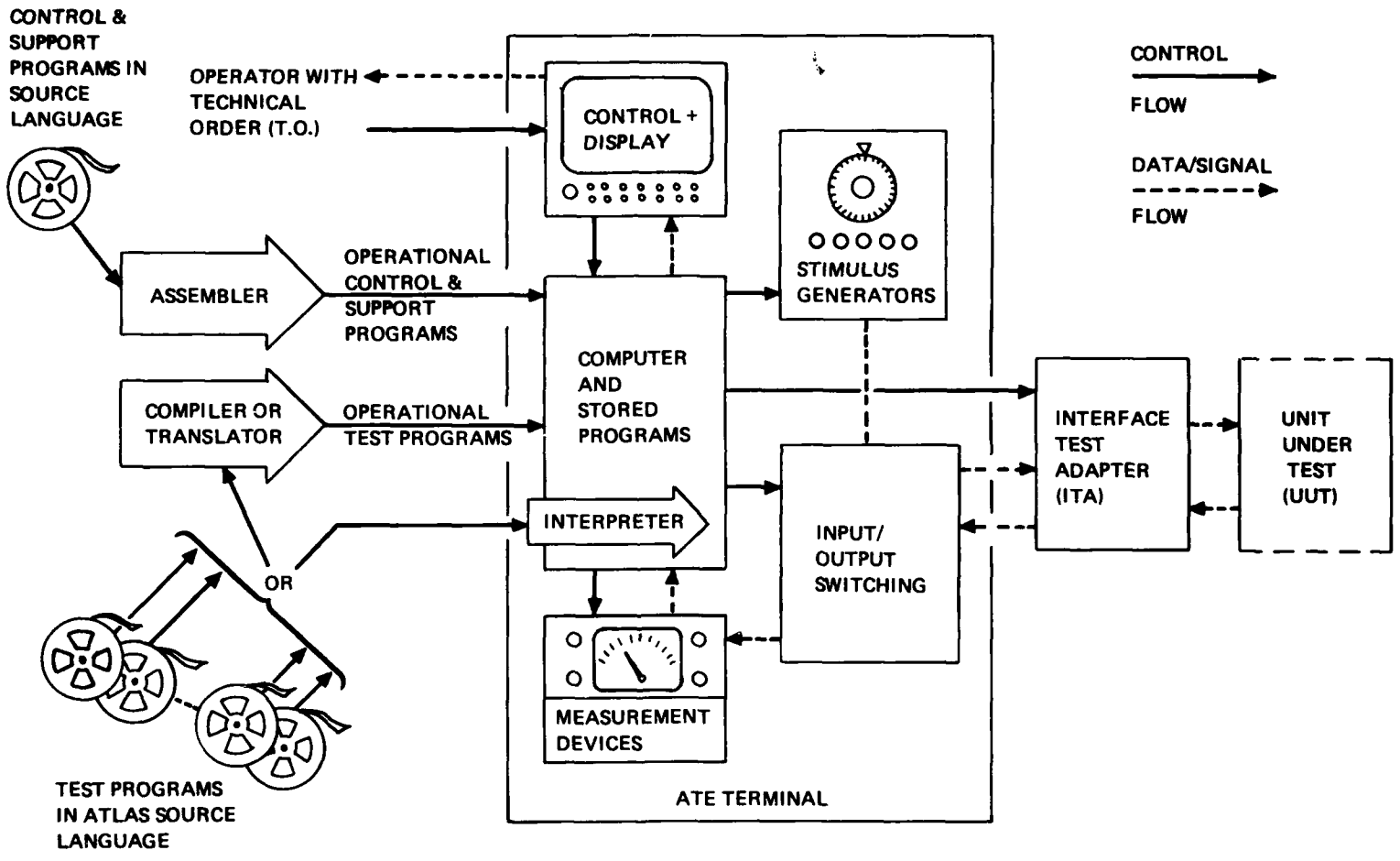


Figure 1.3-2. Typical ATE Configuration

hardware and software phasing and tracking adherence to their respective development milestones. Section 4.0 discusses actual status measuring and reporting mechanisms. Section 5.0 identifies methods of recognizing and correcting problems. Section 6.0 discusses unique ATE and TS considerations relevant to measuring and reporting software status. Sections 7.0 through 11.0 contain a bibliography, guidebook topic vs DOD document cross-reference index, glossary, list of abbreviations and acronyms, and a subject index.

Section 2.0 APPLICABLE DOCUMENTS

Government documents dealing with the topics covered by this guidebook are:

MIL-STD 483, Configuration Management Practices for Systems Equipment, Munitions and Computer Programs.

MIL-STD 1521A, Technical Reviews and Audits for Systems, Equipment and Computer Programs.

AFR 800-2, Program Management

AFR 800-14, Management of Computer Resources in Systems

SSD Exhibit 61-47B, Computer Program Subsystem Development Milestones

Section 3.0 HARDWARE AND SOFTWARE PHASING-MANAGEMENT VISIBILITY

In a ground system such as ATE, operating system software is a part of the system "central core" which in turn is linked with a functional test station, an Interface Test Adapter (ITA) and a UUT test program to comprise a total system. Since the development of these components is inter-dependent (i.e., software design affecting hardware design in hardware), decisions are required to allocate requirements and prioritize developmental work. This section provides guidelines for software scheduling, prioritizing and status monitoring.

3.1 ESTABLISHING PROJECT MILESTONES

Schedules are time phased representations of a plan often displayed in graphic form. They are essential to the successful acquisition of ATE and TS since they represent one of the foundations of effective management. Experience has shown the real key to effective software management involves three essential elements. First, it requires development of a credible plan. The schedule is of course fundamental to this process. Second, a system of measurements and reports is established to determine actual performance against the plan. Finally corrective action is taken when actual performance deviates from the plan. Schedules and their effective control are helpful to the TS and ATE acquisition engineer for the following reasons.

3.1.1 Milestone Accomplishment

Milestones provide a greater degree of reliability that the system will be delivered on time. Critical milestones are established. These milestones reflects events which occur during the life of the system acquisition. Because the acquisition engineer must monitor and evaluate contractor performance and must be aware when his actual performance deviates from the schedule, these milestones should be selected with care. By this is meant the following:

a. Each milestone event should be unambiguous in nature. Schedule milestones, particularly those on which contractual matters are based, are a subject of negotiation during the fact finding and contract definitization phase of the source selection process. This process is described in the Contracting for Software Acquisition Guidebook. During negotiation of these milestones particular attention should be placed on ensuring the milestone event can be recognized as such when it has occurred. For example, a milestone event described as delivery of the software part II specifications leaves little doubt as to what is meant. The term "delivery" implies a contractually prescribed sequence of events which must take place. Little if any doubt exists in determining when these events have occurred. However, a milestone described as completion of software part II specifications can be ambiguous. The term "complete" can be defined in a number of different ways. It can mean the specifications have been written, printed and are waiting for company management to review them. They may not be available to the USAF for sometime to come. Or, it can mean the contractor understands and has defined the specifications but, they have not been written, reviewed or made available to the USAF.

b. Milestones should describe tangible events wherever possible. A milestone stating "analysis complete" is not a tangible event. It is often difficult to determine when this event has occurred. However, if a milestone such as "analysis document" is used, a more tangible event has been described. In the latter case a specific product is produced which can be evaluated. In the former case only a concept, or the state of mine of the analyst is described and it is difficult to determine the precise date that such an event occurs. In general, milestones should be selected

as those points in time at which a specific tangible product such as a software document or test report has been produced.

c. Milestones should occur at significant events. The purpose of a milestone event is to measure partial completion at an intermediate step and use this as a measure of the degree to which the entire job is complete. Therefore, milestones of events which are necessary preliminary steps in completion of the entire software system should be selected. If instead, insigificant events or events which are not in the main stream of software development are chosen, the purpose for milestone scheduling is compromised. For example, the Program Management Plan (PMP), Computer Resources Integrated Support Plan (CRISP), and Test Requirements Document (TRD) are significant events in the development of ATE and TS software. Completion of these activities is a necessary step which precedes coding and checkout of the software. Consequently, availability of satisfactory versions of these provide an indication of the degree to which the entire software job is complete. On the other hand a milestone established at the completion of a software subroutine may not be indicative of the degree of total job completion since the subroutine could be developed independently.

d. Milestones should be placed at intervals which roughly coincide with the intervals of status reporting. To provide an excessive number of sequential milestones all falling due in the same reporting period is meaningless. Further, it can waste contractor and government effort required to define, measure, report and review actual performance against these milestones. If too few milestones are scheduled, there will exist risk that problems will go unreported and both the contractor management and the government may lose control of the software job.

e. The majority of ATE and TS projects involve software of sufficient scope to require a hierarchy, or multiple tiers or schedules. Milestones for these should be identified so as to achieve consistency throughout each tier of a multi-tiered schedule. Each succeeding tier after the highest level divides the same software job into finer degrees of scheduling and reporting detail. Therefore, continuity should be maintained so that it is possible to locate every higher tiered milestone on each succeeding lower tiered schedule. Further consistency in naming of events should be maintained so that lower tier milestone could, if desired, be located with respect to any higher tiered schedule. Care should be maintained to convey the message that each lower tiered schedule is a more detailed plan of the same job reflected on a higher tier schedule rather than appearing as a schedule for a totally different job.

f. The establishment of project milestones makes it possible to accomplish tasks in a more orderly fashion. The very discipline required to think through a complex TS or ATE software job, and define events for purposes of milestone establishment and associated flow times of activities leading to these events is in itself an extremely useful practice. To do this requires definition of every principal software element, every major document, every significant portion of data, and every algorithm by which the input is translated to the output. In short, to do this scheduling process requires the contractor software manager and his acquisition engineering counterparts in the USAF to define all the software performance and documentation. This is the first necessary step in any successful software development activity.

3.1.2 Schedule Control

Schedule control is a primary means of determining potential problems in time to institute effective corrective action. Methods of recognizing and abating such problems are described in Section 5.0. Schedule control is an

essential element in estimating, allocating and planning manpower and other required resources. Government contracts today are requiring ever increasing levels of detail in the evaluation of contractor proposals and in fact finding. In software cost estimating, the primary resource is manpower. Manpower estimates, in turn, are derived from master and lower tiered schedules. Software resource planners must divide the overall task into time phased subtasks compatible with the master schedule and contractual milestones. Unrealistic scheduling can seriously impact the success of the software development effort; particularly, if the required resources (i.e. special skills) are not available when they are needed.

3.2 DEFINING MILESTONE PRODUCTS

In any system acquisition, the contract dictates the Tier 1 milestones which form the "backbone" of all subtier schedules. So called "Tier 1 milestones" usually include, but are not limited to, "contractual milestones." Contractual milestones for software development are twofold: events and deliveries/submittals. Some typical contractual milestones for ATE and TS software acquisition and development are:

a. Events

(1) System Requirements Review (SRR)

(2) System Design Review (SDR)

(3) Preliminary Design Review (PDR)

(4) Critical Design Review (CDR)

(5) Functional Configuration Audit (FCA)

(6) Physical Configuration Audit (PCA)

(7) Formal Qualification Review (FQR)

(8) First Unit Delivery

b. Deliveries/Submittals

(1) Prime Item Development Specifications (PIDS)

(2) Configuration Item (CI)/Computer Program Configuration Item (CPCI) Specifications

(3) Computer Program Development Plan (CPDP)

(4) Configuration Management Plan (CMP)

(5) Interface Control Drawing (ICD)

(6) Test Requirements Documents (TRD)

(7) Qualification Test Procedures

(8) Acceptance Test Procedures

(9) Qualification Test Report

(10) Acceptance Test Reports

(11) User's Manual

(12) Version Description Document (VDD)

(13) Computer Program Media

(14) System Delivery

Beginning with these basic contractual milestones, schedulers should prepare master and subtier schedules. It is recommended that a separate contractor organization exist to prepare and maintain these very important schedules. Usually the contractor organization performing this function is Program Planning and Control (PP&C). Scheduling is a skill that requires a certain aptitude and specialized training. The critical qualification for a good scheduler is common sense. Some highly competent engineers and design managers

are not good schedulers. They are frequently so preoccupied with finding solutions to troublesome design problems, that they do not "see the forest through the trees."

3.2.1 Scheduling

Effective scheduling demands a dedicated and sustained effort and should be performed by a group skilled in schedule preparation and maintenance. This group is responsible for the following tasks.

a. Identification of all required tasks and interdependencies that exist between tasks.

b. Accomplishment of tasks and delivery of output products in an appropiate and logical sequence.

c. Establishment of an accountability system for monitoring completion of all required tasks and delivery of all required output products.

d. Anticipation of resource requirements - neither too little, too late, nor too much too soon, but what is needed, when it is needed, and where it is needed.

3.2.2 Resources

Resources which must be anticipated, quantified, scheduled and provided for are:

a. Time - manhours/manmonths of effort and the arrangement (prioritization) of that effort.

b. Skills - having the required personnel with the requisite skills available when needed.

c. Materials & Facilities - ensuring the availability of adequate program development facilities and peripheral and the communications network available with the required capabilities and capacities.

These resources of necessity cross organization boundaries and must be under the control of the scheduling organization. It is reemphasized that milestone definitions should be tangible entites. Functional organizations contributing to the overall schedule objectives should clearly understand what is needed at a given point in time and the interfacing organizations using a given organization's input must clearly communicate what is needed.

3.2.3 Communications

The establishment and maintenance of these needed communications is established as follows:

a. Schedule Planning and "Networking"

b. Schedule development

c. Schedule Statusing

d. Management Review

e. Problem Identification and Abatement

The following subsections describe each of the above major steps in schedule development and monitoring in synoptic fashion. Subsequent sections of the guidebook elaborate on the detailed techniques of schedule development and monitoring including considerations unique to ATE and TS software.

3.3 SCHEDULE DEVELOPMENT AND MAINTENANCE

An effective schedule status function, properly staffed and fully supported by management will not necessarily prevent a project from experiencing schedule problems. However, the advantage of such systems is that potential problem areas are highlighted at an early stage when effective corrective action can be implemented. An effective schedule/status discipline is difficult to implement for software development projects because it takes time, dedicated resources and

reporting commitments from personnel unaccustomed to this type of discipline. However, if goals are realistic (and time is allocated to programmers to provide schedule information), the project manager will find the process well worth the effort. The following are the basic steps in estabishing and maintaining software schedules.

3.3.1 Schedule Planning and Networking

Starting with need dates and backing up from there, the first step the software project manager must undertake is the planning of a schedule or building a network. The "network" concept is a method for representing a logically sequenced schedule plan in graphic form. The steps necessary in developing a network are:

a. Develop a statement of work (SOW) of what must be done.

b. Identify tasks, activities, outputs required to achieve the end objective.

c. Relate interdependencies of tasks.

d. Assign responsibilities.

e. Assign estimated completion dates (ECD's).

Figure 3.3-1 illustrates a simplified development network for a given CPCI. A complete network would contain an organizational or personnel assignment for each activity or task. At this point, no mention will have been made of the amount of time necessary to accomplish each activity, since at this point the scheduler is only concerned with defining and sequencing events. Next the project manager must allocate the available time between the present date and the final job completion date into intervals appropriate for each subtask. Work normally performed in a serial fashion may have to be done in parallel. Next, the interdependencies of these tasks must be evaluated. For example, the question, "do modules A and B really have to be done prior to beginning design at module C?" must be answered. Perhaps schedule difficulties can be partially alleviated by parallel development efforts or possibly some of the design effort may be consolidated. Finally, upon completing a realistic planning network, the project manager should assign personnel responsibilities to each activity to insure adequacy and quantity of skills to accomplish this preliminary plan. The project manager should review this plan with his staff to insure a clear understanding of each programmer's responsibilities. Wherever possible the outputs of each plan element should be tangibly defined. The graphical presentation of the charted plan should, for simplicity, contain only a brief but succinct statement descriptive of that activity. If further explanation is required, each applicable event may be flag-noted and appropriate job descriptions can be written on a separate chart. Since the network is the "blueprint" for the formal schedule, it is extremely important to review and obtain commitments from those parties responsible for executing each activity/task to insure that interdependencies are correct and manpower allocations are realistic.

3.3.2 Schedule Development

This activity involves translation of the network into a formal schedule. The network is usually prepared by the project manager using a "window" of time available to do the job to meet a required completion date. Time network is translated into various tiers of schedules by the scheduling (PP&C) organization. The actual resultant schedule is an annotated chart depicting start and completion dates for significant events for each development task. It depicts parallel and serial activity annotated with flag-notes and legends which are used as warnings to management reviewers of actual or potential impact of schedule slides or missed milestones. Figure 3.3-2 depicts two tiers of simplified scheduling for a typical ATE system/software development task. It is seen how increasingly greater detail of

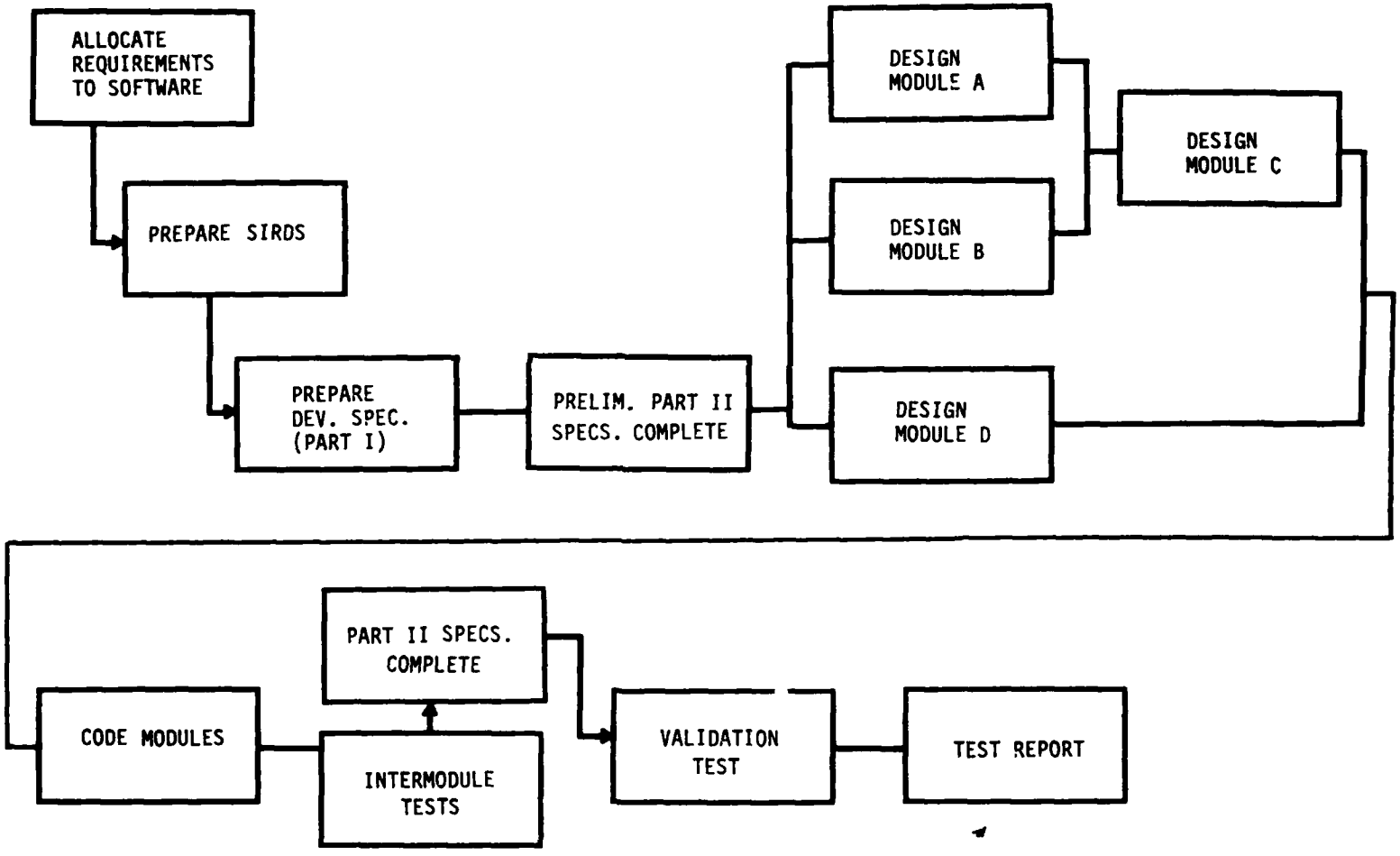


Figure 3.3-1. Software Development Network - CPCI XYZ

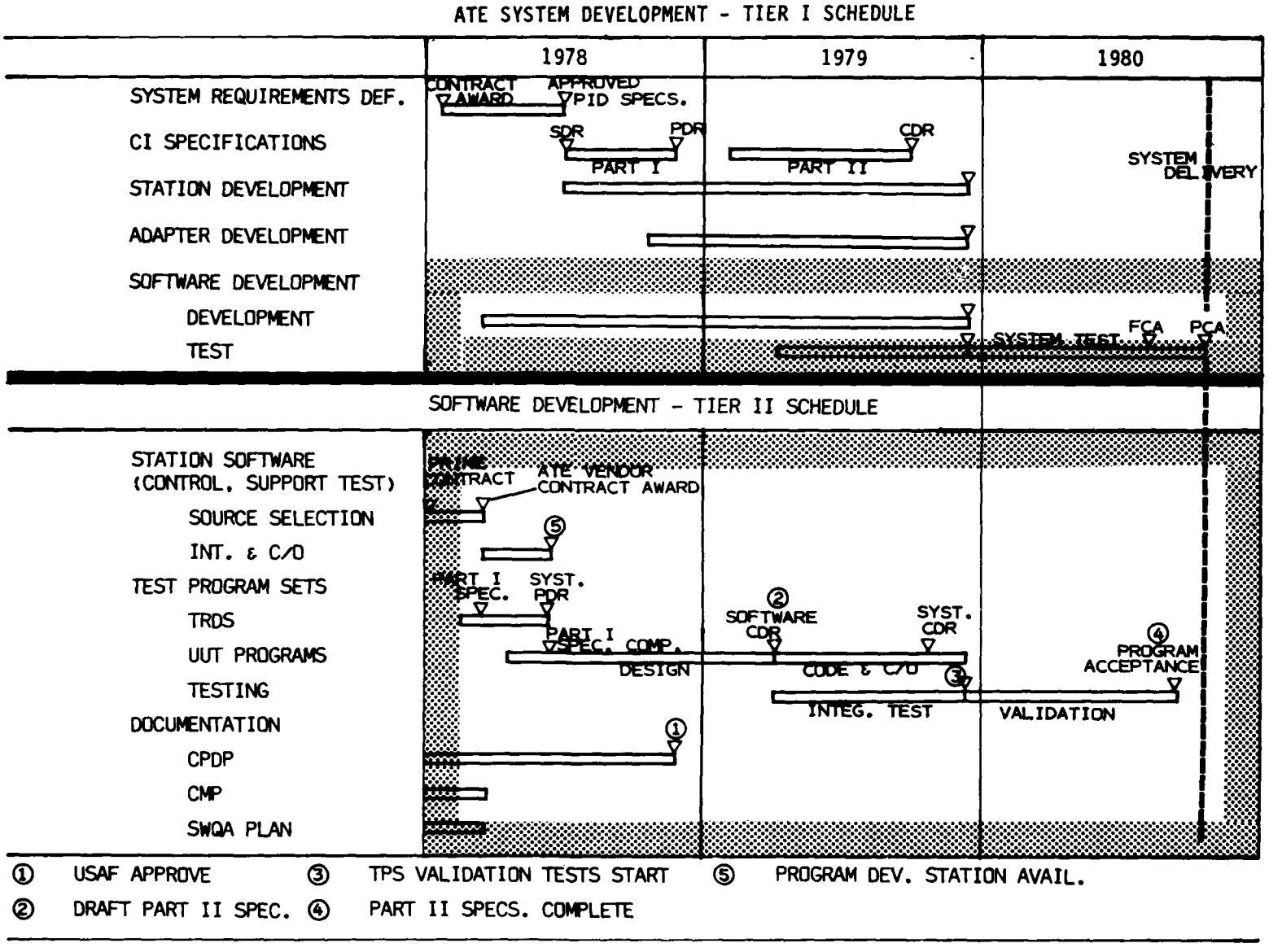


Figure 3.3-2. Tier I/Tier II Schedules

required events is exposed in the Tier II schedule for the general tasks shown in the Tier I. Several points should be considered in associating milestones with schedule tier levels:

a. A milestone representing a software task generally fall into one of four tier levels based on the significance of the event scheduled and the level of detail associated with the respective tier.

b. The process of scheduling milestones is best performed from the top down (similar to the top-down design process), from high-level project milestones to low-level detailed milestones.

c. Scheduling from the top down facilitates the task of ensuring that schedule dates for detailed milestones are consistent and compatible with higher level commitments. This becomes an increasingly difficult task as project size and complexity increase.

Management schedule review systems for controlling development of the entire ATE or TS system will typically employ four tier levels of schedule monitoring as follows:

a. Tier I - contract and/or major program milestones of the type that would appear on a master schedule. Examples could be "requirements definition complete," "detailed design complete," or "system installation complete."

b. Tier II - USAF interface milestones that represent the transmittal of major output products either to or from the USAF, the completion of major reviews, or the approval of budget or schedule for a succeeding project phase.

c. Tier III - system/subsystem integration and test milestones that generally represent the completion of activities above the detailed computer-module level but below the system level.

(1) One example is the functional subsystem test of a group of related computer modules.

(2) Tier III milestones may represent critical events constraining the development task but not directly part of it. Examples are conversion of data from old systems, restructuring of old data bases, implementation of new software or hardware critical to the project, implementation of distributed processing, and/or tele-processing network system additions or deletions.

d. Tier IV - detailed milestones at the individual computer-module level or at the individual data base design level. Use of standard milestones is a particularly effective method for measuring progress toward completion through a common frame of reference for:

(1) Individual computer-module design, code, and test.

(2) Individual data base and development.

For ATE and TS software a set of tiered schedules should be prepared for tracking each CPCI or major software component. In the case of ATE software, however, the vendor supplied portions of the control, support and test programs do not warrant such tracking unless major modifications are required to be made by the contractor or the vendor. When all CPCI's are scheduled with appropiate detailing through the various tiers, the initial schedule development job is complete.

The formality and attention paid to maintaining these schedules on an up-to-date basis cannot be overemphasized. Apathy toward schedule maintenance and adherence can destroy management visibility and control of project. The degree of formality in managing schedules is dependent on the size, duration and complexity of the project. Some of the

schedule maintenance, display and monitoring techniques in use today are:

a. Project dedicated control rooms

b. Schedule Planning Displays

c. Tiered Schedule Displays

d. Standard Milestones

e. Milestone status reporting

f. Late item reporting

g. Program Management Networks

The use of these schedule management tools is discussed in the following paragraphs.

3.3.3 Schedule Statusing

Once having developed a complete and realistic schedule and coordinated with all concerned functional organizations, it must be vigorously reviewed, critiqued and updated. This process called schedule statusing measures actual accomplishment against commitments, reveals potential schedule problems and creates problem abatement planning. Fig. 3.3-3 describes some typical symbology for depicting schedule adherence.

In managing schedules for complex projects, contractor management will typically employ various color coded legends for highlighting impending schedule impact. For example the solid portion of a bar chart designating "activity complete" may be color coded red if the activity is actually delinquent, yellow if potentially delinquent or "black" if "on schedule." When properly annotated with appropriate symbology, footnotes, color coding, etc., the schedule should be formally constructed on some large, back lighted panel or projected display so that it may be reviewed and critiqued by the project management. Usually a control room specially constructed for this purpose (referred to in this guidebook as a Management Information Center or MIC) is used to hold periodic "Schedule Performance Reviews." These meetings are attended by all responsible project managers and supporting personnel.

While the master and lower tiered schedules alert management to a potential problem, further detail is required to comprehend the seriousness of the problem and to pinpoint where recovery effort must be placed. The software "Worm" chart (see Fig. 3.3-4) provides visibility into percent complete as well as providing management an overview of "rate of progress." The slope of the curve hints at whether behind schedule conditions are recovering or deteriorating. Use of such charting techniques are further discussed in Section 4.0 for both percent complete and technical performance progress reporting.

3.3.4 Management Review

Management review is the process by which key project managers evaluate project progress. Tracking status against milestone commitments is a critical component of schedule control discipline. The schedule board must remain fully aware of milestone status in order to effect any work resequence/reschedule actions necessitated by performance problems. The various schedule conditions into which management must categorize activity and chart performance are:

a. Estimated completion date

b. Potential slippage

c. Actual slippage

d. Completed

e. Cancelled

f. Suspended

The key challenges in schedule management are:

a. Realistic milestone establishment

b. Accurate reported status

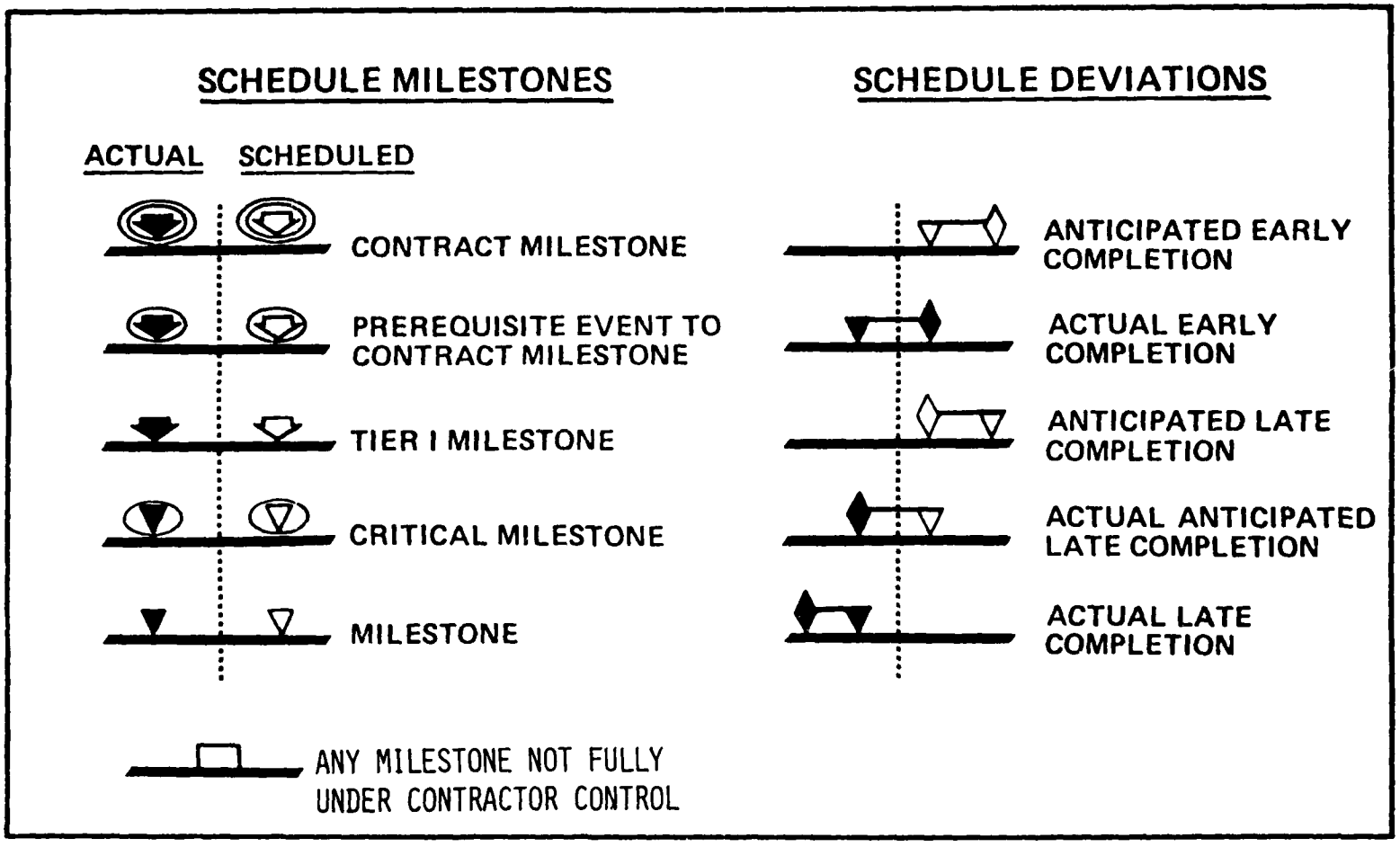


Figure 3.3-3. Schedule Symbology

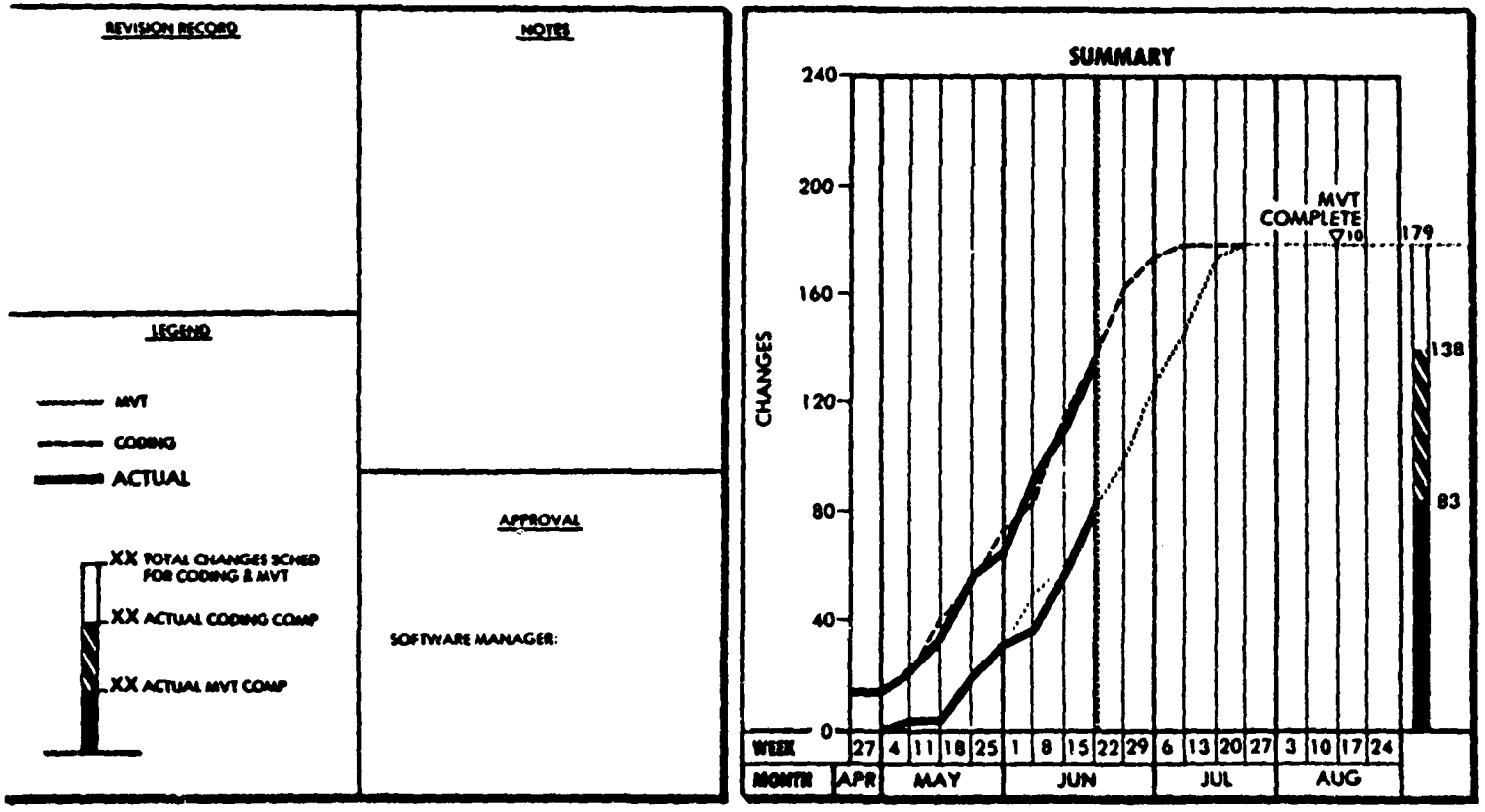


Figure 3.3-4. Software "Worm" Chart

c. Verification of "actuals" through published evidence of completion

Oftentimes, although initial work tasking estimates are derived in good faith based on available data, pressure to meet those commitments is felt by the committing organization. There is a tendency to either "shortcut" the originally scheduled effort or delete subtasks not visible on the higher tiered schedules. The key to verifying that a milestnoe is in fact complete is in defining clearly and completely the output products that are generated to satisfy the milestone requirement, then ensuring that each output product, when completed, meets the established criteria. The responsible organization cannot be expected to perform this function impartially, since there will be a built-in bias in favor of the quality of the output product. The recipient organization should not be expected to be impartial either, since the bias will tend to be the reverse, an output product may never quite measure up to what is expected.

An effective procedure is to establish an independent output product release monitor to:

a. Verify and validate the quality and completeness of output products.

b. Secure output product acceptability signatures from responsible and recipient organizations as required.

c. Publish an official record of the output products that have met the established release criteria.

d. Transmit the output products in a secure and timely manner to the recipient.

Software Quality Assurance and Configuration Management organizations play a vital role as "output product release monitors" by ensuring through design surveillance and review activities that quality specifications and product descriptions are released to define the design. See guidebook on Software Quality Assurance. Schedule reviews by management should be conducted with scrupulous regularity. Presenters must be asked to point out differences in status from the previous review and evaluate impact, if any. Responsible project managers should review schedules prior to formal presentation to higher management to identify problem areas and report recovery plans. Such recovery plans should be simple, but complete and formal and should address:

a. Milestone impacted

b. Schedule milestone date

c. Statement of Problem (reason for slide)

d. Recovery action plan

e. Impact after implementing recovery plan

f. Project approval signatures

Section 4.0 describes reporting and review techniques applicable to ATE and TS software. Detailed problem abatement methods are discussed in Section 5.0.

Section 4.0 STATUS MEASURING AND REPORTING

This section summarizes the types of indicators that are identifiable and suitable for measuring software development status. In addition, there is a discussion on various reporting methods for comparing the status against the schedule. These reports provide input to both contractor and Air Force management for recognition of areas of risk or concern.

4.1 STATUS MEASUREMENT PARAMETERS

Excluding cost which is the subject of the Cost Measuring and Reporting Guidebook, there are three basic parameters which measure software status - percentage of completed code, technical performance compliance, and documentation release. These measure how much is done; how well the software works; and how current are the required engineering releases. Once these parameters are determined, they are provided for management review. The following subsections discuss how to gather, organize, and present the data generated through these parameters.

4.1.1 Percentage Complete

In tracking completion percentage, a scheme should be adopted which measures each Computer Program Component's (CPC's) progress against predetermined goals as committed by previously submitted estimates. Each CPC (module, subroutine, program unit, etc.) consumes a portion of each major development effort - preliminary design, detail design, code and checkout, and test and integration. As the schedule develops, estimates of percentage of overall effort that each CPC will take are evaluated and established. Percentage of completed code is the ratio of completed lines of code to estimated total required lines of code, expressed as a percentage. It is used to estimate degree of completion of a coding task. These estimates are used to form the basis for estimating current status vs. the schedule for purposes of management review.

An example of an effective CPC monitoring technique is the unit development folder (UDF). The UDF concept is a standard for documenting plans and progress of a CPC. The UDF is used by the contractor in developing each CPC. It is also applicable to TS and ATE software development performed by the USAF in an operational environment. The programmers are the most common users of the UDF with the software/ATE or TS Program Manager reviewing them at prescribed intervals. The USAF acquisition engineer should specify in the contract a procedure similar to the UDF. A large project will require a very detailed breakdown of each CPCI into many CPCs, each with a UDF, for development or modification. It provides the best available information concerning the real, up to date status of a programmer's activity. The information contained in the UDF constitutes the kind of tangible evidence of job status needed to maintain effective management control of the software.

The programmer responsible for design of a CPC begins by identifying each element or subroutine needed to complete the CPC. Each CPC is named, and the programmer then prepares a UDF for each, with a cover sheet (or equivalent) which sets a schedule for the completion of design, coding, checkout, and technical documentation. As each of these detailed tasks is completed, the results are placed in the UDF along with the date actually completed and review signatures are entered on the UDF cover sheet. Figure 4.1-1 contains an example of a typical UDF cover sheet. Examples of the results which may be included in a UDF are: text, flowcharts and subroutine lists for completion of design; current listings of main program and subroutine compilations and load maps for completion of coding; written verification of program performance, written conformation of operational functioning, output verification for technical performance and copies of draft and final documentation for technical documentations.

CPC # _____ (Title) _____

		Date Completed	Date Reviewed
1.	Descriptive Text Complete	_____	_____
2.	Flowcharts Complete	_____	_____
3.	Subroutines Identified	_____	_____
4.	Program Coded and Compiled	_____	_____
5.	Program Loaded with Dummy Subroutines	_____	_____
6.	Program Debugged with Dummy Subroutines	_____	_____
7.	Program Loaded with Subroutines	_____	_____
8.	Functional Interaction Verified	_____	_____
9.	All Operational Modes Functioning	_____	_____
10.	Output Verified	_____	_____
11.	Draft Documentation Complete	_____	_____
12.	Final Documentation Complete	_____	_____

Figure 4.1-1. Example of UDF Cover Sheet

The UDF then becomes the mechanism whereby a programmer can easily review his task schedule, report completion of tasks, and record acceptance of results. The degree to which the UDF is complete provides the best visibility of the "percent" of completion. It establishes naming and referencing conventions for the products of CPCs which make these products more accessible to other project personnel. Communication between the individual assigned a particular series of CPCs and his superior is more efficient (and less costly) since tangible results are available for objective and pertinent review. Using the information from the cover sheet from the UDFs, a software development progress report for each CPCI can be generated as in Figure 4.1-2. The contractor should have a form of the progress report tailored to his requirements. Air Force representatives must know the content of the progress report used since it might be displayed during a program review as discussed in paragraph 4.2.

Looking at the Development Progress Report, there are three percentages displayed. The first is the % category in the far left hand column. This value is an estimate of percent of the total effort each CPC requires. With this percentage, a manager is provided a numerical representation of the importance of each CPC.

The second percentage appears across the report opposite the CPC/Module Name. This is an estimate of the percentage of the effort for a CPC that the associated development requirement takes. These values will vary depending upon the project specifications. The development requirement percentages result from coordination by the Air Force acquisition engineer and the contractor. Some of the considerations which influence these values are the size and complexity of the CPCs, the number and size of the subroutines for each CPCI and the number of new concepts to be developed requiring more design development than normal. The last percentage is the % work complete in the last column on the right. As each requirement for the CPC is completed, it is approved by management on the UDF and checked off on the development progress report. Figure 4.1-2 contains an example of a CPC titled DEMOPLBK and indicates it is completed through PROGRAM LOADED WITH SUBROUTINES signified by the checks in appropriate columns. Taking the percentages from the top of each column and totaling them, the value of percent work complete in this example is 55%. Thus the progress report shows visually each CPC, its importance, and completeness for review of the program manager and, if desired, by the USAF acquisition engineer.

During development from program start through PDR and up to CDR, the various CPCIs are defined with descriptive text and flowcharts designed for identified CPCs. These products are very difficult to define in terms of partial completion due to variables of size and importance changing during development. Therefore, only full completion is normally used to accurately determine their status.

In measuring coding and checkout, several functions are involved. Some of these include program coding, compiling, loading with dummy subroutines, debugging with dummy subroutines, and finally loading with the functional subroutines. Additionally, if a large number of subroutines are required, coding and compiling of these is required. One way to estimate partial coding completion is to compare the lines of code to the flowchart. Using the percent of the flowchart completed as a guideline, an estimate of completed code can be determined on a one to one basis. If one fourth of the flowchart is coded, it may be assumed one fourth of the coding is complete.

4.1.2 Technical Performance Monitoring

This process involves measuring and evaluating the degree to which the evolving ATE/TS software meets the requirements established for it. The primary require-

CPCI #012 ADVANCED TRAINING DEVELOPMENT PROGRESS REPORT

% CATEGORY	CPC/MODULE NAME	DESCRIPTIVE TEST COMPLETE 10%	FLOWCHARTS COMPLETE 10%	SUBROUTINES IDENTIFIED 5%	PROGRAM CODED AND COMPILED 10%	PROGRAM LOADED WITH DUMMY SUBROUTINE 5%	PROGRAM DEBUGGED WITH DUMMY SUBROUTINE 10%	PROGRAM LOADED WITH SUBROUTINES 5%	FUNCTIONAL INTER-ACTION VERIFIED 10%	ALL OPERATIONAL MODES FUNCTIONING 5%	OUTPUT VERIFIED 10%	DRAFT DOCUMENTATION COMPLETE 15%	FINAL DOCUMENTATION COMPLETE 5%	% WORK COMPLETE
4%	PREPROG	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓			80
2%	PPUTIL	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		95
2%	PRINIT	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	100
8%	DEMOPLBK	✓	✓	✓	✓	✓	✓	✓						55
3%	FORMFLY1	✓	✓	✓	✓									35
3%	FORMFLY2	✓	✓	✓	✓									35

Figure 4.1-2. Example of Development Progress Report

ments with which the USAF acquisition engineer should be concerned are those reflected in the Required Operational Capability (ROC), Request for Proposal (RFP), contractor's technical proposal and, beyond all else, the ATE/TS system specification and CPCI specifications. Methods by which periodic reviews of technical performance is monitored is included in paragraph 4.2.

The primary source of information by which technical performance is monitored by the USAF acquisition engineer is contractor supplied status information in the form of documentation, data and test results. Technical performance parameters are included in paragraph 4.1.2.2.

4.1.2.1 Milestone Schedule Status Parameters. Having established a framework and methodology for maintaining the schedule baseline for a software development project, a procedure must be developed for tracking milestone schedule status against committed schedule dates.

Milestone schedule status is the condition of a milestone in regard to actual performance toward its scheduled disposition. One of the most difficult concepts to communicate relative to the scheduling/schedule status function, is the difference between a schedule date and a status date (just as there is a difference between a committed schedule date and a planning or target date). Each is governed by a separate set of rules and conventions. The two dates may be, and frequently are, equal to one another (in an "on schedule" condition, that is), but they do not have the same meaning.

People tend to think in terms of each scheduled milestone having but one date and that the latest estimated completion date (status date) is a "recovery schedule" date that permits the original schedule date to be completely ignored. Project managers will save themselves considerable trouble if they ensure that their project personnel have a common understanding of the scheduling/schedule status terms and definitions in use on their particular project.

Milestone schedule status is expressed in terms of two components: (1) a condition and (2) the date on which the milestone condition was or will be attained. Status conditions are:

a. Estimated completion (E)

b. Potential slippage (P)

c. Slippage (S)

d. Accomplished (A)

e. Cancelled (X)

f. Suspended (Y)

g. Deleted (D)

(1) Estimated completion (E)

(a) The anticipated date for completion of a milestone as provided by the responsible organization.

(b) This E-date must be in the future relative to the current status cutoff date, otherwise it is invalid. The status cutoff date (or status-to-be-reported-as-of date) is the calender date through which status must be reported during the current reporting cycle.

(c) E-dates are provided by the responsible organization based upon its capability to perform and do not require negotiation with or the approval of other parties.

(d) To establish an E-date as a new schedule date, negotiation between affected parties is required in the schedule.

(e) If the reported status date is equal to the schedule date, and the reported status condition is "E," the

milestone is in an on-schedule condition and retains the status condition of "E" for performance reporting purposes.

(f) This condition can exist only when both the schedule date and the estimated completion date are in the future with respect to the status cutoff date; for example, where the status cutoff date is 07-08-78 and both the schedule and estimated completion dates are 07-11-78.

(g) If the status (estimated completion) date is earlier than the schedule date, and both are in the future with respect to the status cutoff date, the milestone is in an "ahead of schedule" condition, and the milestone retains the status condition of "E." An example of an ahead-of-schedule condition would be one where the status cutoff date is 07-08-78, the schedule date is 07-15-78, and the estimated completion status date is 07-11-78.

(2) Potential Slippage (P)

(a) If the reported estimated completion date is later than the schedule date, and both are in the future with respect to the status cutoff date, the milestone is in a potential slippage condition, and the reported status condition of "E" is converted to "P" for performance reporting purposes.

(b) As an example of a P-condition, assume that the status cutoff date is 07-08-78, the schedule date is 07-15-78, and the reported estimated completion status date is 07-22-78.

(c) Also, if a schedule date is in the future with respect to the status cutoff date, and the status date is not applicable, milestone status will be in a P-condition.

(3) Slippage (S)

(a) If the reported estimated completion status date is later than the schedule date, and the schedule date is earlier than (prior to) the status cutoff date, the milestone is in an S-condition; the reported status condition of "E" is then converted to "S" for performance reporting purposes.

(b) An example of an S-condition is where the status cutoff date is 07-08-78, the schedule date is 07-01-78, and the reported estimated completion status date is 07-15-78.

(c) Further, if a schedule date is in the past relative to the status cutoff date, and the status date is not applicable, status of the milestone will be in an S-condition.

(4) Accomplished (A)

(a) When a reported milestone accomplishment/completion/actual has been verified, the reported accomplishment date and status code of "A" for the milestone are confirmed.

(b) The A-date is verified when the established output product(s) or officially accepted "surrogate" evidence of completion (e.g., transmittal letter, signed and approved meeting minutes, document release form etc.) is confirmed to be available, complete, and correct as required. Thus, an A-condition reported by a responsible organization must be verified in the manner defined for that particular milestone.

(5) Cancelled (X), Suspended (Y), or Deleted (D)

(a) Cancellation (X) action occurs when it has been officially determined that a once-valid milestone requirement no longer exists.

(b) Suspension (Y) action occurs when work is officially stopped (held in abeyance), usually as the result of USAF direction.

(c) Deletion (D) action occurs when it has been established that a milestone has originated in error; that is, a valid milestone requirement never existed.

4.1.2.2 Technical Performance Parameters. The parameters associated with measurement of the technical integrity of ATE/TS software are less well defined than those associated with cost, schedule and documentation status. Here the technical proficiency of the acquisition engineer becomes extremely important. In many cases technical integrity is a matter of technical judgement.

In TS, for example, which of several numerical integration schemes is selected may be purely a matter of technical judgement. However, all are not equal in performance. Therefore, to evaluate numerical integration schemes applicable in any given situation requires knowledge of calculus of finite differences and error analysis. If called upon to exercise judgement in technical disciplines outside his skill area, the acquisition engineer should seek consultant assistance. Beyond this, however, technical integrity of ATE/TS software can be evaluated by means of the following parameters:

a. Specification requirements (Section 3.0). The contractor is obligated to provide a system meeting these requirements. Each and every requirement contained in Section 3.0 of the ATE/TS system and CPCI specifications constitute a technical evaluation parameter.

Paragraph 4.2 provides examples of means by which contractors evaluate their own performance against these requirements. The acquisition engineer should insure that reports reflecting performance against each and every such requirement is reported on a regularly occuring basis and that the form of the information is satisfactory for his evaluation purposes.

b. Specification verification requirements (Section 4.0). These constitute the means by which each Section 3.0 requirement is measured in order to verify that the ATE/TS meets this requirement. Therefore, each Section 4.0 specification item is an important technical measurement parameter. Test reports analysis documents and inspection reports provided by the contractor are analyzed by the acquisition engineer to ensure whether the required verification has been demonstrated.

c. Specification Applicable Document requirements (Section 2.0). This specification section defines the MIL-STDS, etc., which are applicable to the ATE/TS system. Therefore requirements included in the corresponding MIL-STDS and other forms of command media constitute technical performance parameters.

4.1.3 Documentation Status

As the entire development proceeds, many documents are generated and accumulated for inclusion in the UDF. The USAF acquisition engineer should require formal release of the required documents as they are available. As discussed in the Guidebook Computer Program Documentation Requirements, the items listed in Figure 4.1-3 are the documents needed for proper USAF utilization of the software. These are the documents, the status of which, the acquisition engineer is primarily concerned with. The means by which this status is obtained is included in paragraph 4.2.

The Program Manager is charged with tracking the status and progress of documentation. It is his responsibility to confirm that the documents produced will meet quality standards on which he and the Air Force have concurred, in order to satisfy the needs. Since the Program Manager is operating in a cost controlled environment, and document production is a significant cost item, he must have visibility of the progress of various documentation activities, to effectively control costs and achieve his profit objectives.

The document standards employed in software development projects have as their objective the production of pertinent documentation, i.e., documents tailored to the needs of the Air Force and pro-

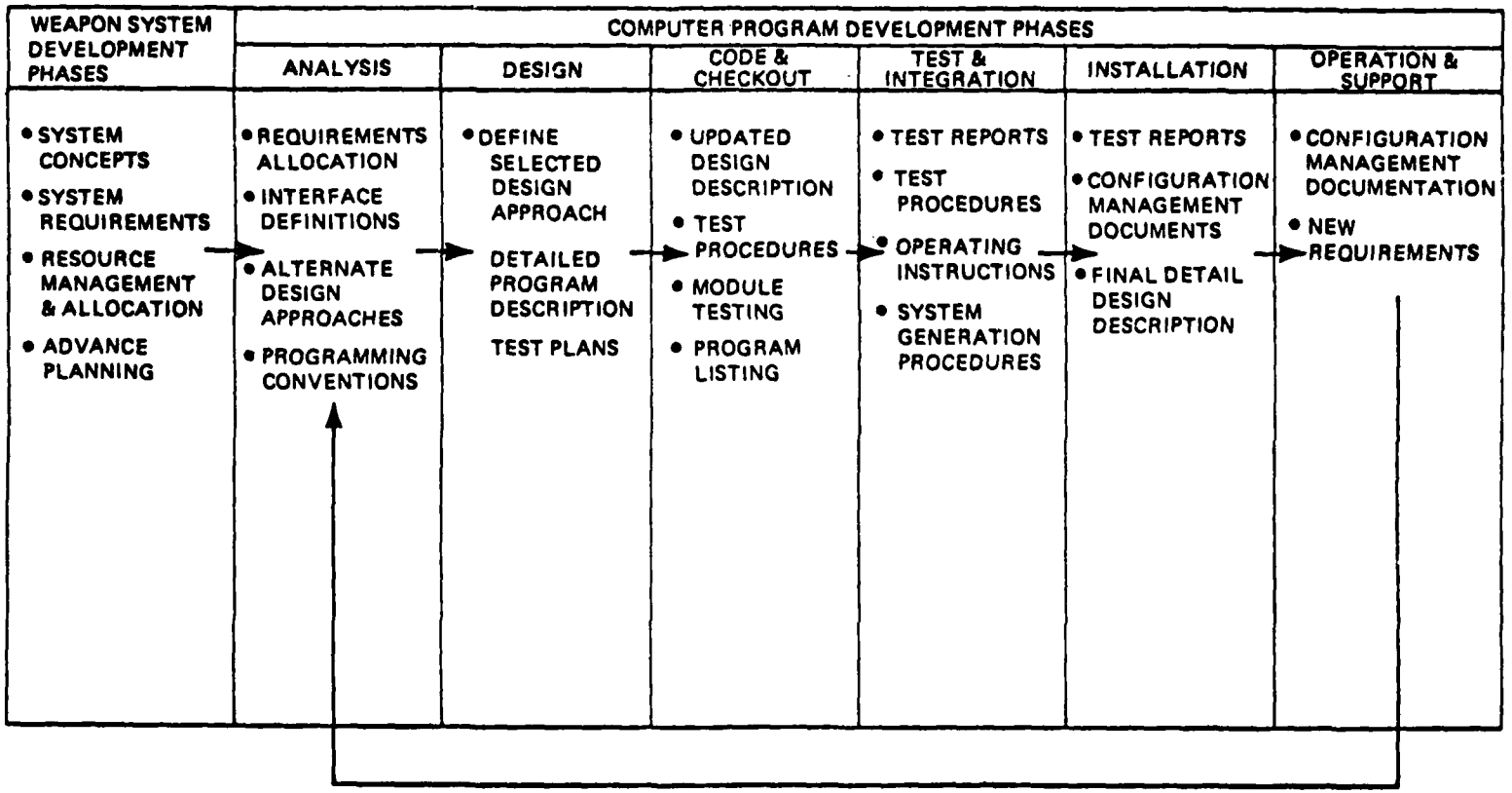


Figure 4.1-3. Documentation Needs in the Computer Program Development Cycle

duced early enough in the development activity to provide necessary visibility (particularly to the USAF). This implies that documents must be planned to be produced in parallel with other development tasks (usually in parallel with design activities), rather than planned as the last obligation of the software developer. Document standards are developed and applied which directly address the functional needs of specifically-identified audiences. Furthermore, documents are submitted in draft form to the USAF acquisition engineer for early review.

The standards described here are not universal; rather, they are project unique, shaped by contractual obligations, USAF needs, and project objectives. The specific document plans and standards employed may vary from project to project, but they exhibit the general characteristics discussed herein. These new document standards may somewhat increase the costs associated with review and packaging. The materials are now more critically reviewed to ensure that they meet the USAF's needs, and multiple documents are being packaged for publication instead of a single, omnibus Maintenance Document. However, having a specific definition of form and content of a document, and an understanding of the document's intended use, the programmer should find the preparation task less burdensome and time-consuming. Further, key information such as input/output formats, design specifications for components and sub-routines, and standard nomenclature conventions is recorded and made easily available to allow the programmer and integrator greater visibility of how the individual portions of a software capability relate to each other. This visibility should help each individual to perform his assigned task with increased efficiency and less cost.

4.2 STATUS REVIEW

This section describes the review processes required by USAF command media, both for contractor and government use, and those in use primarily by contractors which have arisen as a result of experience with a number of ground systems.

4.2.1 Contractor Internal Reviews

The purpose of contractor internal reviews is to assess on a continuing basis the ATE or TS software in terms of task results. The results reviewed include plans, documents, software elements, data and supporting materials. In addition these reviews are conducted to assess performance, including quality of the software and of the personnel actions associated with it.

Internal review conducted by contractors is a continuing process. While the specifics vary between contractors normally the techniques employed have the characteristics described below.

4.2.1.1 MIC Facility. A physical room, referred to herein as the Management Informantion Center (MIC), is set aside in which computer program status information is permanently displayed. Normally one or more individuals are assigned the responsibility to collect periodic status information from the software manager and maintain the MIC information in a current status. The contractor organization performing this activity is normally the Program Planning and Control (PP&C) function. However, it is fundamental to the management philosophy associated with MIC activities that, while PP&C may physically collect and display software status information, the ATE or TS software manager is responsibile for the accuracy and validity of information displayed therein. For this reason this manager normally acknowledges the information by placing his signature on the displayed information after PP&C has prepared/reviewed the displays.

4.2.1.2 MIC Function. The information displayed in the MIC concerns all important attributes of the status of the TS or ATE and is normally updated weekly. Figure 4.2-1 contains an example of MIC display material associated with the

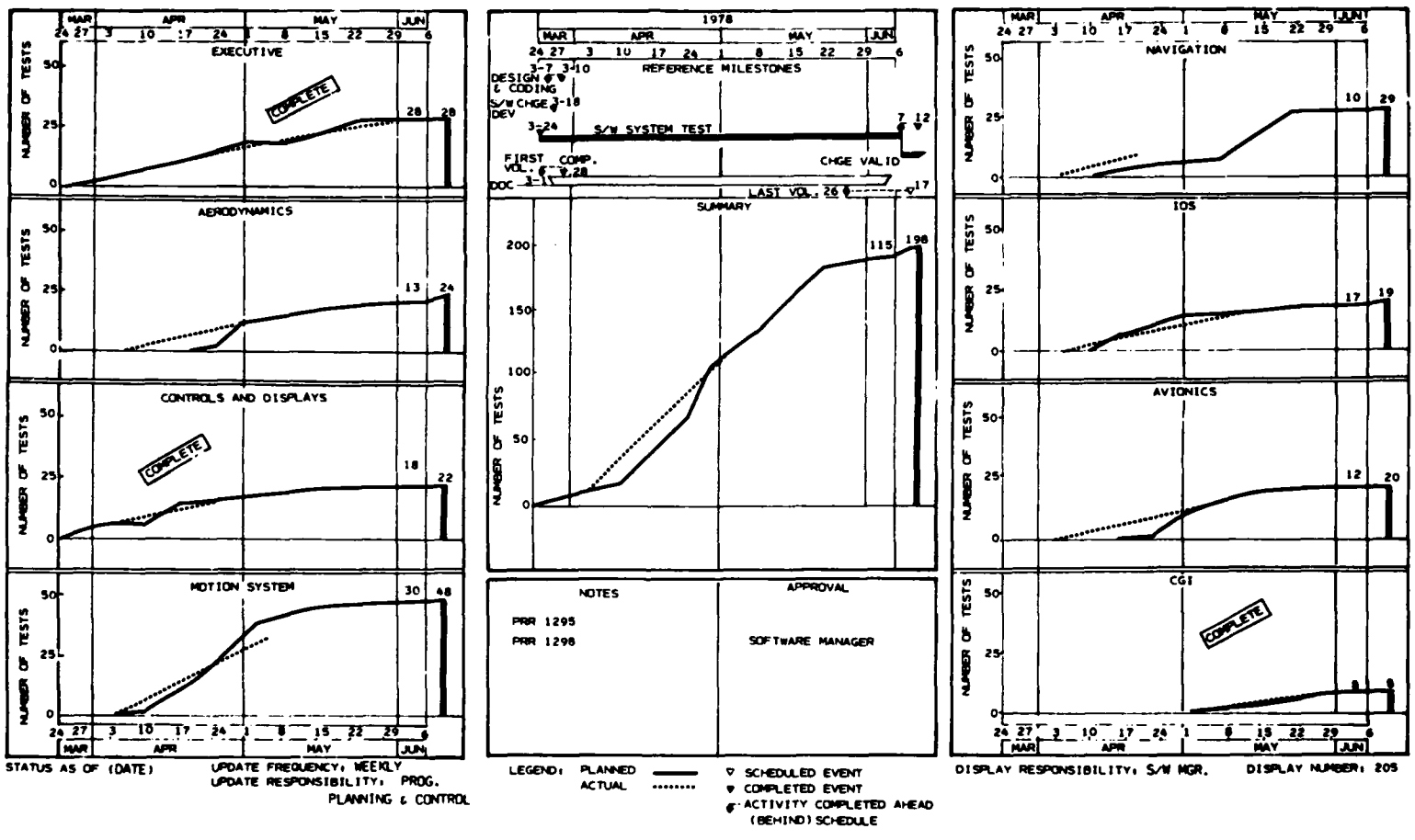


Figure 4.2-1. Management Information Center (MIC) Display Example

software test status for a typical TS. Here the parameter measured, and status displayed, is the number of software certification tests successfully completed versus the number scheduled.

In general the contractor will select, for MIC display, information depicting status of each and every performance parameter which the contractor is obligated, by virtue of his contract, to demonstrate as a condition of USAF acceptance of the TS or ATE system.

The program manager is obligated, by virtue of the ATE or TS system specification, and each software configuration item (CPCI) specification, to verify performance. These parameters, therefore, constitute his contractual obligation. Figure 4.2-2 contains a hypothetical MIC room display example for a training simulator whose partial requirements are as follows:

a. A separate configuration item exists for the TS computer software and the Computer Generated Imagery (CGI) system software.

b. The specifications require that the CGI software occupy not more than 50% of the central processing unit (CPU) capacity and not more than 75% (24.4K) of the main storage capacity. It further requires that not more than 4.65 Mega bytes of disk storage capacity are to be used.

c. The specifications require that the TS software use not more than 90% of the CPU and not more than 30.3K words of the 32K main storage capacity.

Evidently the following events are reflected on the TS Main Memory Utilization status chart, Figure 4.2-2.

a. The requirements, as reflected in the applicable specification, changed as a result of two Engineering Change Proposals (ECP). While the original specification allowed the contractor to use only 29.7K words of main memory this was relaxed to 30.1K as a result of ECP 270 and further relaxed to 30.7K by ECP 320.

b. The contractor, during the period starting early 1976 and continuing for more than a year had considerable difficulty meeting this criteria. However, as a result of corrective software redesign actions taken in mid 1977 he was able to meet the requirement and, as of the date of this information, the TS software performance meets its memory utilization design capacity.

Note, also, on Figure 4.2-2, that the specification verification method is evidently different for the two computer systems. While the TS computer capacity must be verified by actual measurement in some prescribed form, the CGI software is verified by analysis.

Figure 4.2-3 provides hypothetical examples of other parameters which may be displayed in a MIC environment. These charts depict the status of coding and module verification (by test) of changes being incorporated in a set of TS test software.

In general there are an almost endless variety of chart and display formats which appear in a MIC display. However, the general requirements are summarized as follows:

a. The display should depict the important (specification requirement) parameters simply and consistently. This aids understanding and comprehension of the displayed data.

b. It should be possible to visually scan the information and determine whether the status is "good" or "bad" without requiring a detailed knowledge of the specification requirement or of the software CPCI. This makes it possible for contractor and government personnel of widely varying backgrounds to understand the information.

c. The charts should be sufficiently simple that they can be readily compre-

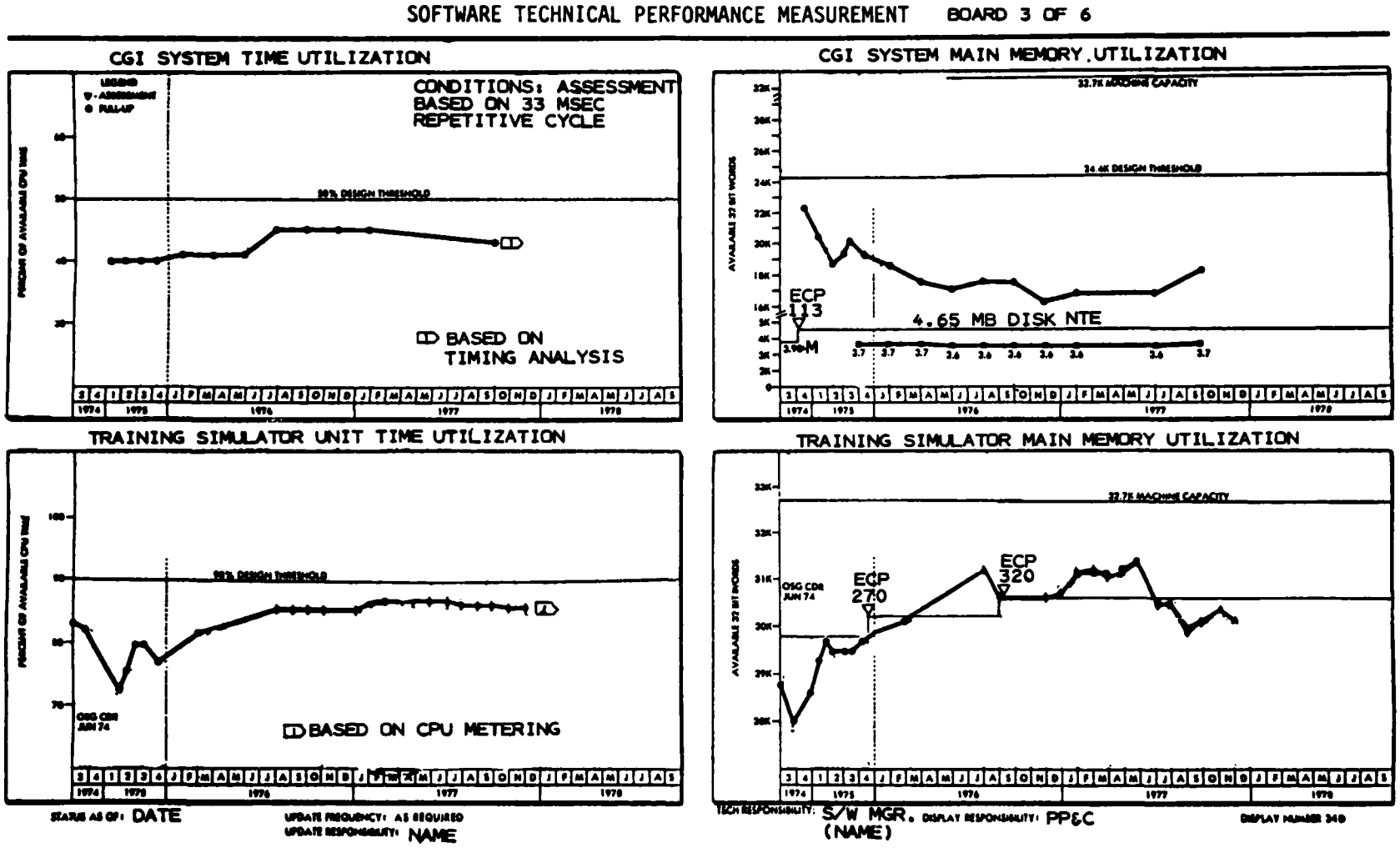


Figure 4.2-2. Technical Performance Measurement

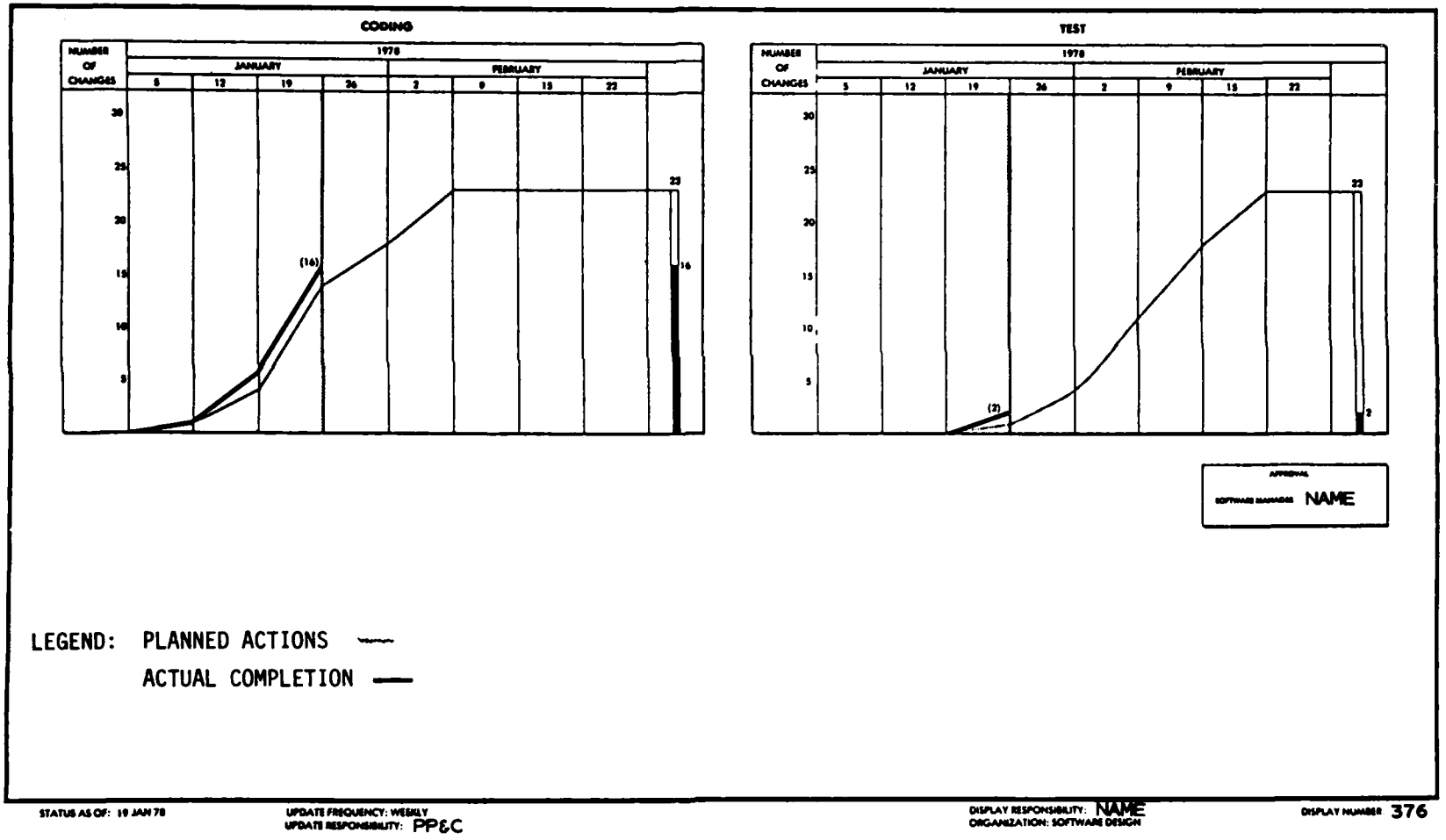


Figure 4.2-3. Contracted Changes - Automatic Test Equipment - Atlas Program XXXX

hended in reduced size. The examples given were taken from actual MIC boards, each containing two or more charts, which are approximately 3 ft x 4 ft in size. Yet, the information reduced to the 8 1/2 x 11 inch format of the figures is still legible.

This makes it possible for the contractor to distribute MIC information to individuals not having convenient access to the MIC room. Further, it is a convenient form for distribution of MIC contents to the USAF Systems Program Office (SPO).

4.2.1.3 MIC Advantages. Normally the MIC is available for review by all TS or ATE program key personnel. Therein, fully visible, is displayed all problems so that coordinated action by responsible managers is directed. In most successful programs the MIC room is a work room where program staff meetings are held and where problems are discussed, actions assigned and performance reviewed.

In actual practice on most programs the MIC room, if properly developed and maintained is occupied by the Program Manager and his staff a very significant portion of the time. It is, in a very real sense, the control center of a contractor's program activities.

4.2.1.4 MIC Alternatives. Although the MIC is a contractor function, and is used by contractors whether or not a MIC is specified in his contract, it is recommended that ATE or TS acquisition contracts be written in such a way that the contractor is obligated to provide facsimile copies of all MIC information to the SPO on a periodic (i.e., monthly) basis.

In this way the ATE or TS acquisition engineer has the best and most up to date information available to him on the important aspects of the evolving software and hardware system. Further, this information provides an excellant source of discussion items at periodic Program Management Reviews (PMR) held between the contractor and the USAF.

If the contractor is not obligated to maintain a MIC and has no intent of maintaining one for his own use then it is recommended monthly reports, reflecting actual status vs planned status as follows:

a. Status concerning and deliveries/ submittals indicated in paragraph 3.2, should be reported. In particular, the acquisition engineer should ensure that the contractor is required to monitor his actual vs planned performance, and report this to the USAF, for every Section 3 requirements item and every Section 4 verification item in the ATE/TS System Specification. In addition, items selected by the acquisition engineer from CPCI specifications should also be reported.

b. Information reported should, like those examples indicated in paragraph 4.2.1 above, contain technical requirement performance as will schedule and cost performance, since all three elements are necessary to the successful software acquisition engineering management job.

c. Specific information necessary to support USAF requirements as reflected in MIL-STD-1521A (See paragraph 4.2.2), should be required of the contractor.

4.2.2 Government Reviews

MIL-STD-1521A, Technical Reviews and Audits for Systems, Equipments and Computer Programs is applicable to TS and ATE.

4.2.2.1 Review Types. MIL-STD-1521A specifies seven separate government reviews as defined below. Normally these reviews are held at contractor facilities. Since the contractor and SPO/ acquisition engineer are geographically

separated, the contractor should be required to submit review material well in advance of the review meeting.

The acquisition engineer's role in these reviews is extremely significant; he is the government's primary representative, ensuring the ATE/TS system, when delivered, meets USAF requirements. To carry out this function adequately he must be armed with knowledge of the software requirements as reflected in the Request for Proposal (RFP), bidding contractor's proposal and the specifications. It is therefore recommended that the acquisition engineer carefully refamiliarize himself with these documents before attending any of the government reviews.

The following paragraphs in this section are based on MIL-STD-1521A and indicate items which the acquisition engineer should review and evaluate during conduct of these meetings. It is recommended the acquisition engineer, using these as a "checklist," compile a list of primary requirements in precise quantitative or qualitative obtained from the RFP, technical proposal, ROC, specifications (if available) and any other available information, and take this list with him to government review meetings.

During these meetings the contractor's job is to present information concerning the evolving ATE/TS or other ground system. The government's job is to review and evaluate the information and then make a judgement concerning the degree to which the government's requirements are being met. The list of requirements should be used to evaluate, point by point, the contractor's ATE/TS software system. During this evaluation the contractor should be required to discuss, or demonstrate, how the evolving software meets each requirement contained in the list.

As a word of caution, it should be pointed out the acquisition engineer is frequently at a disadvantage at these meetings. This stems from the fact that the contractor's role places him in the position of controlling the information to be presented. Secondly, since the contractor is designing the system, he normally has greater knowledge of his design than the acquisition engineer.

This inherent disadvantage can be offset by two things. First, as previously indicated, the acquisition engineer should come with his own information and knowledge by the method discussed above. Secondly, he should persist in his efforts to require the contractor to demonstrate evidence that his system design meets each and every requirement. The contractor should be persuaded to describe not only how his system works, but how his system meets the requirements.

a. System Requirements Review (SRR). The objective of this review is to ascertain the adequacy of the contractor's efforts in defining system requirements. It is conducted when a significant portion of the system functional requirements has been established.

b. System Design Review (SDR). This review is conducted to evaluate the optimization, correlation, completeness, and risks associated with the allocated technical requirements. Also included is a summary review of the system engineering process which produced the allocated technical requirements and of the engineering planning for the next phase of effort. This review is conducted when the system definition effort has proceeded to the point where system characteristics are defined and the CI are identified.

c. Preliminary Design Review (PDR). This review is conducted for each CPCI or aggregate of CPCIs to (1) evaluate the progress, technical adequacy, and risk solution (on a technical, cost, and schedule basis) of the selected design approach, (2) determine its compatibility with performance and software requirements of the CPCI development specification, and (3) establish the existence and compatibility of the physi-

cal and functional interfaces among the CPCIs and items of equipment, facilities and personnel.

d. Critical Design Review (CDR). This review is conducted for each CI when detail design is essentially complete. The purpose of this review is to (1) determine that the detail design of the CI under review satisfies the performance and engineering requirements of the CI development specifications, (2) establish the detail design compatibility among the CI and other CIs, facilities, computer programs and personnel, (3) access producibility and CI risk areas (on a technical, cost, and schedule basis), and (4) review the preliminary specifications.

e. Functional Configuration Audit (FCA). A formal audit to validate that the development of a CI has been completed satisfactorily and that the CI has achieved the performance and functional characteristics specified in the functional or allocated identification.

f. Physical Configuration Audit (PCA). A technical examination of a designated CI to verify that the CI "As Built" conforms to the technical documentation which defines the CI.

g. Formal Qualification Review (FQR). The test, inspection, or analytical process by which products at the end item or critical item level are verified to have met specific procuring activity contracual performance requirements (specifications or equivalent). This review does not apply to requirements verified at FCA.

4.2.2.2 Review Functions. Figure 4.2-4 illustrates the life cycle events associated with acquisition of ATE and TS software and indicates the points within this life cycle where formal government reviews take place. In addition, it indicates topics which should be reviewed by the SPO acquisition engineering staff at these formal milestones. Topics to be discussed at these reviews are indicated below.

a. The SRR is normally conducted during the system conceptual or validation phase. Such reviews may be conducted at any time but normally are conducted after the accomplishment of functional analysis and preliminary requirements allocation to ATE/TS computer program CIs to determine initial direction and progress of the contractor's system engineering management effort and his convergence upon an optimum and complete configuration. Since, for these systems software cannot really be separated from hardware, both disciplines should be reviewed. The total system engineering management activity and its output should be reviewed for responsiveness to the SOW and TS/ATE system requirements. Representative items to be reviewed should include the results of the following:

(1) Requirements Specification (See the Requirements Specification Guidebook).

(2) Functional Flow Analysis, including total ATE/TS software functional flow diagrams.

(3) System/Cost Effectiveness Analysis.

(4) Trade Studies (e.g. addressing ATE/TS system functions in hardware/firmware/software).

(5) System Interface Studies, such as the CGI interfacing with the motion system for a TS or the contractor furnished software, adaptors, etc. with purchased ATE hardware/software.

(6) Generation of Specifications.

(7) Program Risk Analysis.

(8) Integrated Test Planning.

(9) Producibility Analysis Plans

(10) Technical Performance Measurement Planning

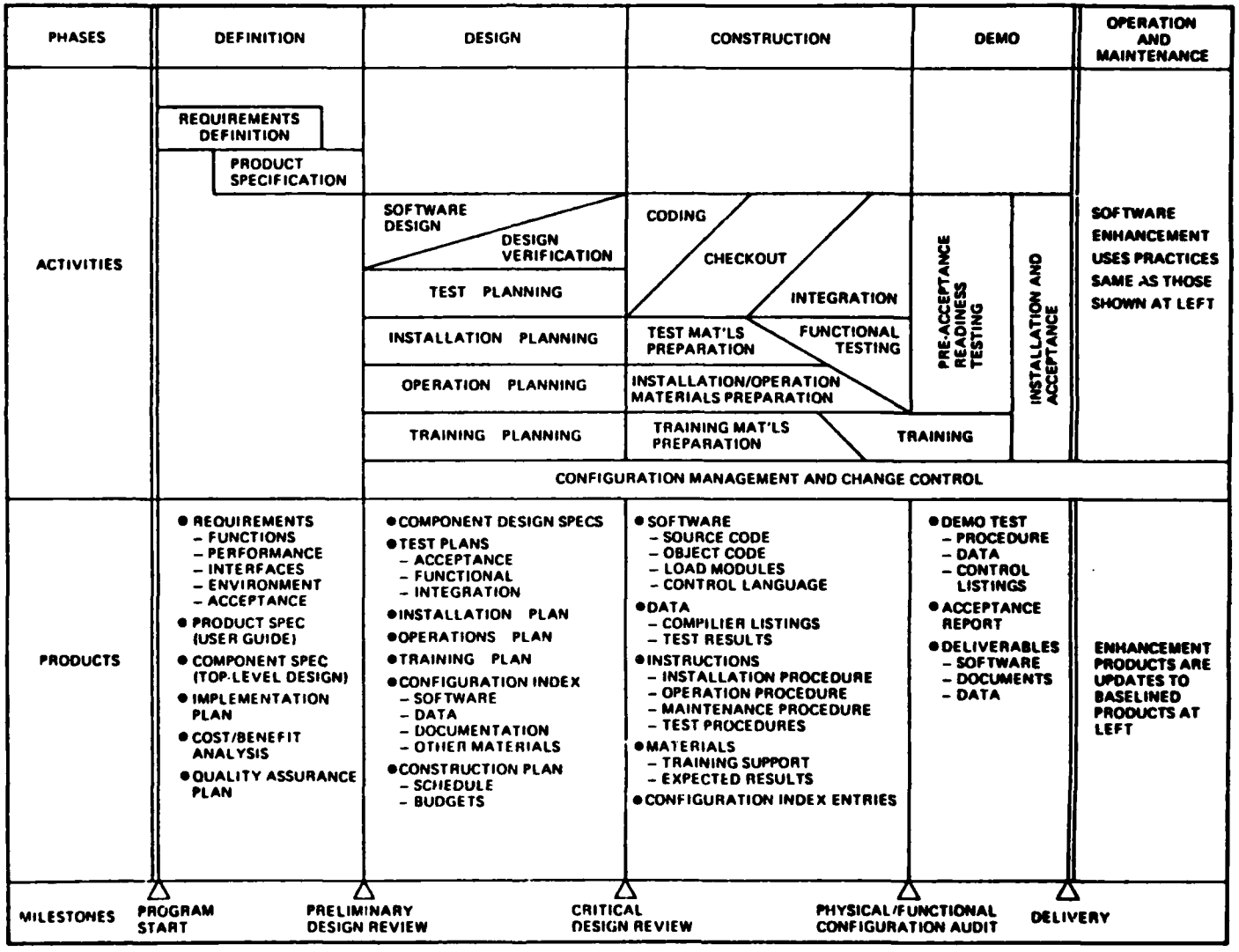


Figure 4.2-4. ATE and TS Software Life Cycle

(11) Engineering Integration

(12) Data Management Plans

(13) Configuration Management Plans

(14) Human Factors Analysis

(15) Life cycle cost analysis

The contractor should describe his progress and indicate problems in:

(1) Risk identification and risk ranking.

(2) Risk avoidance/reduction and control.

(3) Significant trade-offs between ATE/TS system or system segment specification requirements/constraints and resulting engineering design requirements/constraints and logistic/cost of ownership requirements/constraints and unit production cost/design-to-cost objectives.

(4) Significant hazard consideration should be made here to develop requirements and constraints to eliminate or control these system associated hazards. While the ATE/TS software is not normally involved in system safety analyses it can be used to improve system safety problems.

(5) Information which the contractor identifies as being useful to his analysis and available through the procuring activity should be requested prior to or at this review (e.g., prior studies, operational/support factors, cost factors, test plan(s), etc.). A separate SRR may be conducted for each of the software CIs depending upon the nature and complexity of the ATE/TS.

After completing the SRR, the contractor publishes and distributes copies of review minutes. The procuring activity officially acknowledges completion of the SRR.

b. The SDR is conducted to evaluate the optimization, traceability, correlation, completeness, and the risk of the allocated requirements, including the corresponding test requirements in fulfilling the ATE/TS system or system segment requirements (the functional configuration baseline). The review encompasses the total system requirements as well as the ATE/TS software. A technical understanding is reached on the validity and the degree of completeness of the following information:

(1) System Specification

(2) CPCI Specifications

(3) The engineering/cost of the system

An SDR is conducted for ATE/TS systems which are sufficiently complex to warrant the formal assessment of the allocated requirements (and the basis of these requirements) before proceeding with the preliminary design of CIs. The SDR is primarily concerned with the overall review of the operational/support requirements, updated/completed system specification requirements, allocated performance requirements, and the accomplishment of the system engineering management activities. The purposes of the SDR are to:

(1) Insure that the updated/completed system specification is adequate and cost effective in satisfying USAF requirements.

(2) Insure that the allocated software requirements represent a complete and optimal synthesis of the system requirements.

(3) Insure that the technical risks are identified, ranked, avoided, and reduced.

(4) Ensure that a technical understanding of requirements has been reached and technical direction is provided to the contractor.

The SDR includes a review of the following:

(1) ATE/TS Requirements Analysis

(2) Functional Analysis

(3) Requirements Allocation

(4) System/Cost Effectiveness

(5) Reliability/Maintainability (R&M)

(6) Electromagnetic Compatibility, as appropriate

(7) Software Maintenance Concept

(8) System Safety

(9) Security

(10) Human Factors

(11) Transportability (including Packaging and Handling)

(12) Standardization

(13) Value Engineering

(14) System Growth Capability

(15) Program Risk Analysis

(16) Technical Performance Measurement Planning

(17) Producibility Analysis (i.e., significant aspects of materials, tooling, processes, facilities, skills, etc.)

(18) Life Cycle Costing

(19) Computer Program Development Plan

(20) Design-to-Cost Goals

(21) Environmental Conditions as these apply to software requirements (Temperature, Vibration, Shock, Humidity, etc.)

(22) Results of significant trade studies.

Review Section 4.0 of the ATE/TS system specification and all available software specifications for format, content, technical adequacy, and completeness. All available test documentation, including CI/subsystem and system test plans, are reviewed to insure that the proposed test program satisfies the test requirements of Section 4.0 of the system and Part I CI development specifications. All entries labeled "not applicable (N/A)" or "to be determined (TBD)" in Section 4.0 of the system specification and Part I CI development specification are identified and explained by the contractor.

The following topics should be presented and reviewed for the software:

(1) An overall review of system requirements to assure that a technical understanding of requirements has been reached.

(2) The management controls and methodology that will be used to ensure satisfactory design of computer programs, including the techniques to provide traceability of requirements from the system specification through the computer program development specifications and continuing through the computer program product specifications.

(3) Identification of all CPCIs required throughout the system. Examples are: operational programs; maintenance/diagnostic programs; test/debug programs; exercise and analysis programs; simulation programs, and compilers/assemblers and/or store certification programs and other required support programs (e.g. bootstrap loaders and other tools).

(4) The schedule for the development of each CPCI and the procedure for monitoring and reporting status.

(5) The procedure for monitoring and reporting computer program sizing and timing data and data base storage requirements.

(6) Computer programming standards and conventions to be enforced by the contractor.

(7) Trade-off and design studies that have applicability for decisions relating to:

(a) data base design

(b) computer program language usage

(c) space allocation

(d) operating system and/or executive design

(e) computer instruction set selection

(8) The computer programming techniques to be adopted for use in the system, e.g., on-line processing, off-line processing, parallel or multi-processing, multi-programming, time sharing, etc.

(9) A general description of the size and operating characteristics of all computer programs (e.g., operational programs, maintenance/diagnostic programs, compilers, etc.) to include data base requirements.

(10) A description of requirements for system exercising and identification of functional requirements (exercise configuration, conditions, frequencies, functional simulation, recording, and analysis), and identification of major elements required to implement the exercising capability.

(11) Identification of all computer programming languages to be utilized in the system, and a description of how each language impacts the development, and the operations, maintenance and test areas.

(12) Identification of the computer facilities needed to support computer programs in the deployment phase, and the extent to which these facilities will be provided.

(13) Review of data interfaces with existing automatic data processing systems.

After completing the SDR, the contractor submits copies of review minutes. The procuring activity officially acknowledges completion of the SDR.

c. The PDR is a formal technical review of the basic design approach for a CI or for a functionally related group of CIs. It is normally held after authentication of the Part I development specification(s) and the accomplishment of preliminary design efforts, but prior to start of the detail design.

The contractor presents a review of the following:

(1) Preliminary design of the ATE/TS including its software.

(2) Trade-studies and design studies results.

(3) Interface requirements contained in Part I development specifications and interface control data (e.g., interface control drawings) derived from these requirements.

(4) CPCI development schedule

(5) Value Engineering Considerations, Preliminary Value Engineering Change Proposals (VECP) and VECPs (if applicable)

(6) Description and characteristics of "off-the-shelf" hardware/software including any optional capabilities such as special features, interface units, special instructions, controls, formats, etc.

(7) Existing documentation (technical orders (T.O.), commercial manuals, etc.) for "off-the-shelf" hardware/software and copies of contractor specifications used to procure equipment are made available for review by the procuring activity.

(8) Life cycle costing analysis

CPCI and other non-hardware considerations:

(1) The computer program functional flow should be completed to a flow chart level which allocates the Part I performance and design processing requirements to the individual computer program components of the CPCI.

(2) Storage Allocation Charts. This information is detailed for each CPCI as a whole, describing the manner in which available storage is allocated to individual Computer Program Components (CPCs). Timing, sequencing requirements, and relevant equipment constraints used in determining the allocation are included.

(3) Control Functions Description. A description of the executive control and start/recovery features for the computer program system should be available, including method of initiating system operation and features enabling recovery from system malfunction.

(4) Program Structure. The contractor describes the overall hierarchial structure of the computer program, the reasons for choosing the software modules described, the extent to which top-down development will be used within the constraints of the available computer resources, and any support programs which will be required in order to develop/maintain the program structure and allocation of data storage.

(5) Security. An identification of unique security requirements and a description of the techniques to be used for implementing and maintaining security within the data processing subsystem shall be provided.

(6) Reentrancy. An identification of any reentrancy requirements and a description of the techniques for implementing reentrant routines.

(7) Computer Program Development Facility. The availability, adequacy, and planned utilization of the computer program development facility should be addressed.

(8) Computer Program Development Facility/Operational System. The contractor provides information relative to unique design features which may exist in a computer program component in order to allow use within the computer program development facility, but which will not exist in the ATE/TS to be delivered. The contractor should provide information on the design of support programs not explicitly required for the ATE/TS system but which will be generated to assist software development.

(9) Development Tools. The contractor should identify any special simulation, data reduction or utility tools that are not deliverable under the terms of the contract, but which are planned for use during program development.

(10) Review word lengths, message formats, storage available within the computer, card and magnetic tape/disk formats, timing, and other considerations which were established in the Part I CPCI development specification. At this time, the interfaces between CPCI and hardware CIs shall be defined sufficiently to enable computer program design to proceed independently.

(11) Analyze word formats, card and magnetic tape/disk formats, transfer rates, etc. for incompatibilities. For interfaces with other CIs or CPCIs which

are the responsibility of another contractor or government agency, draft and/or final Interface Control Drawings (ICDs) should be reviewed.

(12) Review all functional interfaces between CPCIs within the system. (A more detailed review of these interfaces at a lower level is conducted at the CDR or at an In-Progress Review prior to CDR).

(13) Review the structure of the CPCI as a whole with emphasis on the following:

(a) Allocation of computer program components to functions

(b) Storage requirements and allocation

(c) Computer program operating sequences

(d) Design of the data base

(14) Analyze critical timing requirements of the system as they apply to the CPCI to insure that proposed CPCI design will satisfy the timing requirements. Review estimated running time given by the contractor for compatibility with timing requirements.

(15) Review the CPCI interactions with the human factors requirements. Review the man-machine interfaces including operator-inserted on-line commands (e.g., format of command statements, switch actions), formats of machine-generated alerts, and initial draft of display and hardcopy output design.

d. The CDR is conducted on each CI prior to start of coding to insure that the detail solutions as reflected in the draft Part II software specification satisfy performance requirements established by the Part I development specification.

The CDR for ATE/TS sofware is a formal technical review of the CPCI detail design. The CDR is normally accomplished for the purpose of establishing integrity of computer program design at the level of flow charts or computer program logical design prior to coding and testing. For less complex CPCIs, the CDR may be accomplished at a single review meeting. The primary product of the CDR is formal identification of specific computer programming documentation which will be released for coding and testing.

Since computer program development is an iterative process, the completion of a CDR for a CPCI is not necessarily sufficient for maintaining adequate visibility into the development effort through testing. Additional Technical Interchanges (TI) or PMRs may be scheduled post-CDR which address:

(1) Responses to outstanding action items

(2) Modifications to design necessitated by approved ECPs of design/program errors

(3) Updating sizing and timing data

(4) Updated design information, as applicable

(5) Development status reports

(6) Results obtained during in-house testing, including problems encountered and solutions implemented or proposed.

Items to be reviewed. The contractor, as a minimum, should review the following:

(1) Draft of complete Part II CPCI specification with exception of instruction listings, etc. which can only be produced after coding the program. In cases where the CDR is conducted in increments, a complete draft Part II may not be made available until the last one is conducted. If conducted in increments the review would be conducted on the detail design of the CPC(s) being reviewed.

(2) Supporting documentation describing results of analyses, testing, etc., as mutually agreed by the procuring activity and the contractor.

(3) System Allocation Document for CI inclusion at each scheduled location.

The contractor should provide information on firmware which is included in "off-the-shelf" equipment or to be included in equipment developed under the contract. Firmware in this context includes the microprocessor and associated sequence of micro-instructions necessary to perform the allocated tasks. As a minimum, the information presented during CDR shall provide descriptions and status for the following:

(1) Detailed logic flow diagrams

(2) Processing algorithms

(3) Circuit diagrams

(4) Block and timing data (e.g., timing charts for micro-instructions)

(5) Memory (e.g., type Read Only Memory (ROM), Programmable Read Only Memory (PROM) word length, size (total and spare capacity)

(6) Micro-instruction list and format.

e. The objective of the FCA is to verify that the CPCIs actual performance complies with its Part I development specification. Test data is reviewed to verify that the ATE/TS software has performed as required by its functional and/or allocated configuration identification.

The FCA for a complex ATE/TS may be conducted on a progressive basis. The FCA is conducted on that configuration of the CI which is representative (prototype or pre-production article is not produced, the FCA is conducted on a first production article.

Prior to the FCA data (for CPCIs to be audited), the contractor should provide the following information to the procuring activity.

(1) Contractor representation (the test manager should be in attendance).

(2) Identification of software items to be audited:

(a) CPCI Identifications

(b) Specification Identification

(c) Current listing of all deviating/waivers against the CI, either requested of, or approved by the procuring activity.

(d) Status of Test Programs to test configured items with ATE (when applicable).

The contractor's test procedures and results are reviewed for compliance with specification requirements.

The following testing information should be available for the FCA team.

(1) Test plans/procedures and available acceptance test plans/during which pre-acceptance data was recorded.

(2) A complete list of successfully accomplished functional tests which pre-acceptance data was recorded.

(3) A complete list of successful functional tests if detailed test are not recorded.

(4) A complete list of functional test required by the specification but yet not performed. (To be performed as a system or subsystem test).

Testing accomplished with the approved test procedures and validated data (witnessed) are sufficient to insure CI performance as set forth in the specification Section 3 and meet the

quality assurance provisions contained in the specification Section 4. For those performance parameters which cannot completely be verified during testing, adequate analysis or simulations should have been accomplished. The results of the analysis or simulations are used to insure configuration item performance as outlined in the specification.

The contractor should provide the FCA team with a briefing for each CPCI being FCA'd and delineate the CI/Subsystem test results and findings for each CPCI. The discussion should include requirements of the development specification that he was not able to meet including a proposed solution to each item, an account of the ECPs incorporated and tested as well as proposed, and a general presentation of the entire CI development test effort delineating problem areas as well as accomplishments.

An audit of the CI/Subsystem Preliminary Qualification Test (PQT) and/or Formal Qualification Test (FQT), plans/procedures should be made and compared against the official test data. The results are checked for completeness, accuracy, etc. Deficiencies are documented and made a part of the FCA minutes. Completion dates for all discrepancies are established and documented.

An audit of the draft/final CI/Subsystem test report is performed to validate that the report is accurate and completely describes the development tests.

All ECPs that have occurred during the program are reviewed to assure that they have been technically incorporated and verified during the development test program.

PDR and CDR minutes should be examined to assure that all findings have been incorporated and completed.

The interface requirements and the testing of these requirements are reviewed.

After completion of the FCA, the contractor distributes copies of the FCA minutes. The procuring activity officially acknowledges completion of the FCA.

f. The PCA is the formal examination of the as-built version of a CI against its technical documentation in order to establish the CPCI's product baseline. After successful completion of the audit, all subsequent changes are processed by ECP action. The PCA also determines that the acceptance testing requirements prescribed by the documentation is adequate for acceptance of production units of a CPCI by quality assurance activities. The PCA includes a detailed audit of specifications, technical data and tests utilized in development of the software and a detailed audit of technical descriptions, flow charts, listings, manual/handbooks for CPCIs. The review includes an audit of the released software documentation and quality control records to make sure the as-built configuration is reflected by this documentation.

The contractor provides the following information to the procuring activity.

(1) Contractor representation (the test manager, as well as software manager, should be in attendance).

(2) Identification of items to be accepted by:

(a) Specification identification number

(b) CI identifiers

(c) Code identification numbers

(d) Computer program identification numbers

(e) Computer program VDD

(f) Current listings, flow charts, analyses and other software supporting documentation

(g) TRDs for ATE systems and ATE test software

(3) A list delineating all deviations/waivers against the CPCI, either requested or procuring activity approved.

The following should be performed by the contractor on each CPCI being PCA'd and evaluated by the acquisition engineer:

(1) Review Part II specification for format and completeness

(2) Review FCA minutes for recorded discrepancies that required action

(3) Review CPC descriptions and flow charts

(4) Review CPC interface requirements

(5) Review data base characteristics, storage allocation charts and timing and sequencing characteristics

(6) Review flow charts for proper entries, symbols, label tags

(7) Compare top-level CPCI flow charts with CPC flow charts

(8) Compare detailed CPC flow charts with coded program for accuracy and completeness

(9) Check positional handbooks, computer user's manuals, and computer programming manuals for format completeness and conformance with applicable data items. (Formal verification/acceptance of these handbooks/manuals should be withheld until system testing to insure that the procedural contents are correct.

(10) Examine actual CI (card decks, tapes, etc.,) to insure conformance with Section 5 of the Part II specification

(11) Cross-check a current listings of instructions with the listing in the Part II specification

g. The objective of the FQR is to verify that the actual performance of a CPCI as determined through test complies with its Part I development specification, and to identify the test report(s)/data which documents results of qualification tests of the CI. The point of government certification will be determined by the SPO and depends upon the nature of the program, risk aspects of the particular CPCI, and contractor progress in successfully verifying the CI requirements. When feasible, the FQR is combined with the FCA at the end of CPCI subsystem testing, prior to PCA. If sufficient test results are not available at the FCA to insure the CPCI will perform in its ATE/TS environment, the FQR is conducted (post PCA) during system testing whenever the necessary tests have been successfully completed to enable CPCI certification. For non-combined FCA/FQRs, traceability, correlation, and completeness of the FQR is maintained with the FCA and duplication of effort avoided.

4.2.2.3 Other Reviews. In addition to those formal government reviews indicated, two types of less formal reviews are normally held.

a. The first of these is the Program Management Review (PMR), held at either the government's or contractors facilities. Its purpose is the review of program status and special problems by key contractor and government personnel on a regular basis. These are normally held quarterly, or more frequently as needs warrant, and the dates, locations and agenda are directed by the SPO. The acquisition engineer's role in these reviews is to assist with identification of agenda items and participating in these meetings as directed by the SPO.

b. The least formal government review is the Technical Interchange (TI). This varies from a telephone conversation

between the acquisition engineer and his contractor counterpart, to formal meetings involving contractor and using command personnel. The purpose of the TI is to exchange information of a technical nature between involved parties. TI's can be requested by any responsible individual, subject to local SPO ground rules and regulation.

Section 5.0 PROBLEM RECOGNITION AND CORRECTION

We have come now to the most important reason for accurate and complete status measuring and reporting, namely-recognizing and correcting problems. The two prime methods of recognizing and correcting software development problems are:

a. Schedule Reviews (Administrative)

b. Testing (Technical)

The best planned and adhered to schedules will not insure project success if the product design is deficient. The project's test program is an integral part of the project's master schedule and, it is the prime mechanism for recognizing technical problems and high risk areas.

Management techniques for recognizing problem and risk areas, categorizing as to impact and severity, documenting the problem, planning and tracking the solution, are discussed in this section. Formally documented mechanisms for elevating problems through contractor management hierarchy are described. Handling routine development problems through normal change processing channels is treated in the guidebook on Configuration Management. This section of this book concentrates on ways in which major program impact problems are recognized and averted.

5.1 RECOGNIZING HIGH RISK AREAS

Hopefully decisions reached early in the system acquisition phase will minimize high risk areas of software development. Nevertheless, some high risk software development problems frequently prevail such as:

a. Faulty or late availability of support software

b. New or frequent changes to performance and interface requirements

c. Core and timing utilizations exceeding original estimates

d. Changes to processor architecture

e. Available facilities for concurrent hardware and software development

f. Complexity of operating system software.

Routine "make work" type of design problems are an inherent element in any software development effort. Most problems in this category are remedied by routine handloads or "patches" to the baseline program. When a major problem arises requiring a major program redesign, a complete recompilation of the program may be required. These major recompilation efforts, if unforeseen and unscheduled, can cause critical program impact. Overtime, additional manpower and facilities may be in vain in attaining recovery. Unforeseen changes to requirements, processor architecture, etc. cannot, in general, be planned for and, as a result, projects will invariably run the risk of slippages. Sometimes problems can be categorized as "out-of-scope" and solutions can be renegotiated including new schedules. Design or administrative deficiencies which necessitate "in-scope" changes to maintain spec compliance are the ones which threaten project success and which we shall discuss here.

5.1.1 Testing

Verification and test strategies planned and documented in the CPDP serve to detect major design deficiencies as early as possible. Well planned test philosophies supplemented by accurate and complete status reporting result in effective problem recognition. Although testing philosophy is more thoroughly discussed in guidebooks on verification and validation, the test program affects methods of reporting status, and hence general principles are discussed here.

5.1.2 "Block" Change Testing

One approach to software testing which aids in early detection of major software or system level incompatibilities is the so called "Block Change" approach. (See the guidebook on Software Configuration Management, Section 5.0, for a thorough discussion of this testing strategy.) In this approach software is developed and tested in progressively maturing "blocks" of accumulated changes and enhancements to the ultimate CPCI. The initially developed components are tested, integrated and tested again for compatibility, and then validation tested as an initial version of the CPCI.

This initially validated version serves as a baseline against which changes can be incorporated in order to support level testing. For example, for TS programs, initial "Blocks" are made available early in system development to support hardware/software compatibility testing. As higher level tests and other sources of performance changes (i.e., USAF directed) are received by the contractor software design organization, they can either be held open until the next recompilation of the program, or they can be incorporated by handload or patch. (Project needs dictate which approach is used.) When the volume of such "patches" against a given "Block" of CPCI versions has reached unmanageable proportions or when changes are too complex to "patch", a new "Block" or program recompilation of reworked source code is produced. This creates the next "Block" or baseline for a given CPCI. The Block approach is primarily designed to ease management of changes and to prioritize development of CPCI components to support the next level of system testing. The approach provides, as a side benefit, an "early warning" system for discovering major performance deficiencies.

In summary, therefore, the "Block" approach serves two purposes. First, it simplifies change management by creating progressively maturing baselines for each CPCI against which work can be prioritized and against which changes can be written and more easily managed. Second, it provides early recognition of major design deficiencies by exercising initial versions of the end item in a system environment early in the development phase to detect problems in areas such as percentage of available CPU time, memory utilization, and interface compatibility.

This approach is most effective on TS software development because of the greater intermodule dependencies and the impact of program directed changes.

The same approach, on a smaller scale, can however be applied to ATE software development, particularly to initial integration of control, support and test programs.

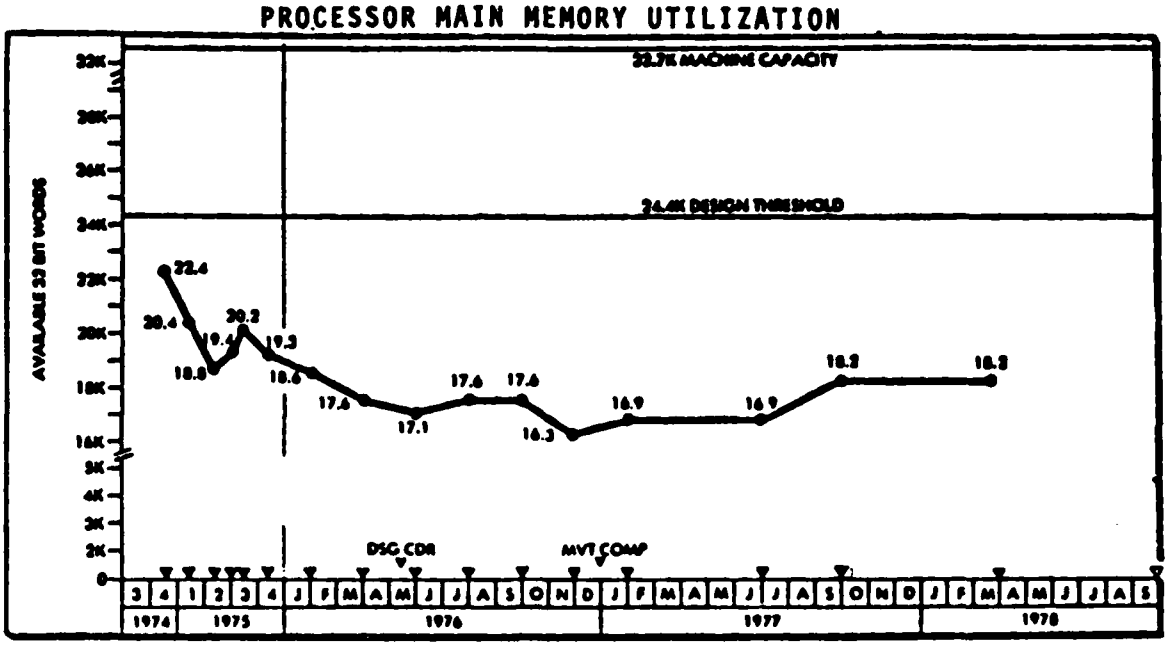
5.1.3 Memory and Timing Utilization Diagrams

Another technique for early recognition of high risk software development problems is the use of CPU memory and timing utilization diagrams. These diagrams help the programmer evaluate his core space allocation requirements vs. remaining available memory. Timing limitations on data transfer or processing operations are similarly critical to program development.

Figure 5.1-1 is a typical memory and timing utilization chart. Use of these or similar techniques provides the visibility needed to track code development against memory and timing capacities. Failure to heed these limitations may necessitate major redesigns.

5.1.4 Requirements Compliance Monitoring

One of the most effective ways of recognizing high risk technical software development problems is through a program of continuous traceability and monitoring of how well test results satisfy



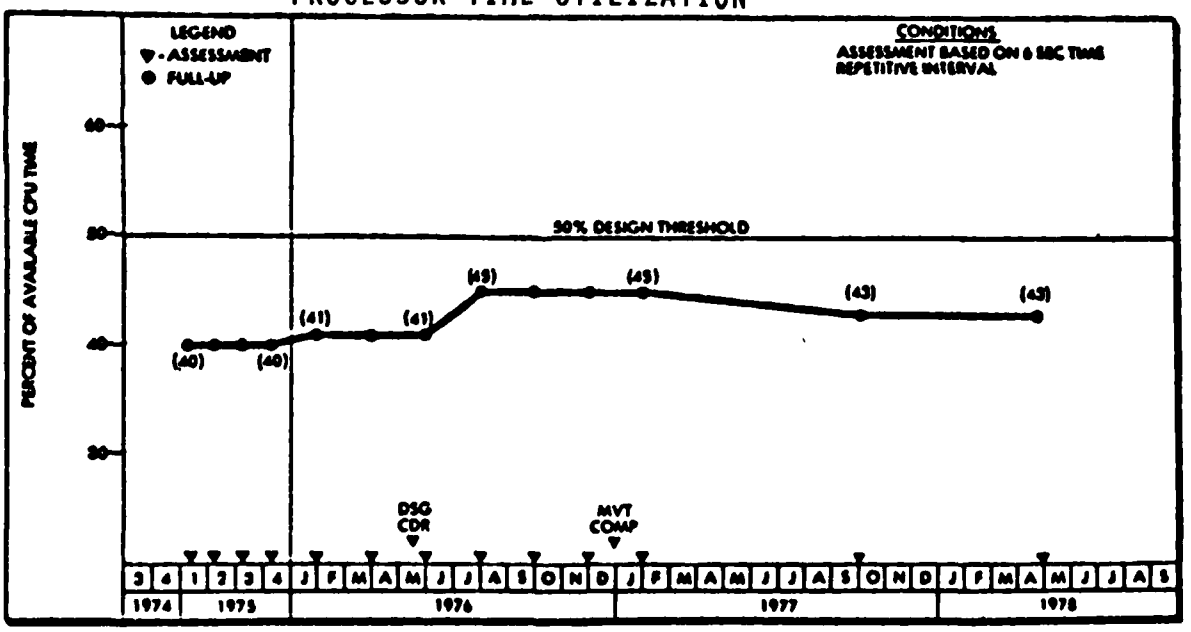


Figure 5.1-1. Memory/Timing Utilization Status Summaries

performance requirements. A well designed test function serves this objective by:

a. Providing an objective evaluation of performance

b. Providing objective evidence of performance verification through analysis or test

c. Providing traceability of performance and test requirements and changes to actual test procedures and results

d. Providing an assessment of how well the requirement was satisfied by the analyses or tests performed.

There are typically 3 levels of testing required to fully check out and "sell off" software:

a. Module verification test

b. Intermodule tests

c. System validation tests

Module and intermodule tests are so-called "single thread" tests. They individually evaluate a given set of functional requirements allocated to modules or interfaces. For example, in TS software, a module designed to simulate the function "conventional gravity weapon ejector" would be tested thoroughly in all of its possible modes or configurations. All logic branches would be executed with all possible combinations of input conditions until the module functioned correctly. Similar tests are run at the intermodule level insuring all interfaces between modules are exercised with all (within reason) probable interface conditions. Finally the integrated program is "validation tested" in a simulated real time system environment during which time all logic paths may or may not be exercised. This final validation test, however, need not execute all available code segments if lower level tests have been properly completed.

Management needs visibility over all 3 levels of testing to insure adequate fulfillment of requirements. Status charts designed to show percentage of tests complete per module together with test results (test reports and assessment summaries) are necessary measures of software development progress. These charts/summaries should be reviewed as part of periodic schedule performance reviews. Equipped with these parameters of progress, project management is now in a position to recognize and abate major schedule impact problems.

5.2 PROBLEM ABATEMENT METHODS

The methods to be employed to plan and execute problem solutions for software vary with project size and complexity. Problems should be worked through a hierarchy of problem reporting and correction mechanisms as a function of the complexity, cost, schedule, number of parties involved, and the degree of agreement which can be reached among the concerned parties. Figure 5.2-1 depicts the a trail of mechanisms for elevating a problem through the ranks until an acceptable solution is reached. The majority of ATE and TS software problems are encountered and resolved at the 3 lowest levels shown in this figure. Typically, when a problem is encountered in design, software test or system test, a Software Problem Report (SPR) (see the guidebook on Software Quality Assurance for a discussion of SPR processing) is processed. A contractor engineering change then embodies the fix in the program design and defines the retest requirements. The SPR is then logged and tracked for discrepancy trends. The foregoing is a routine which results in acceptable solutions to the vast majority of software discrepancies.

Problems unsolvable by simple SPR/Committed Class II change are elevated to a "formal Action Item." These are assigned by the software project manager when immediate action is required to preclude a potential "Area of Concern" or "Top Program Problem" (see Figure 5.2-2)

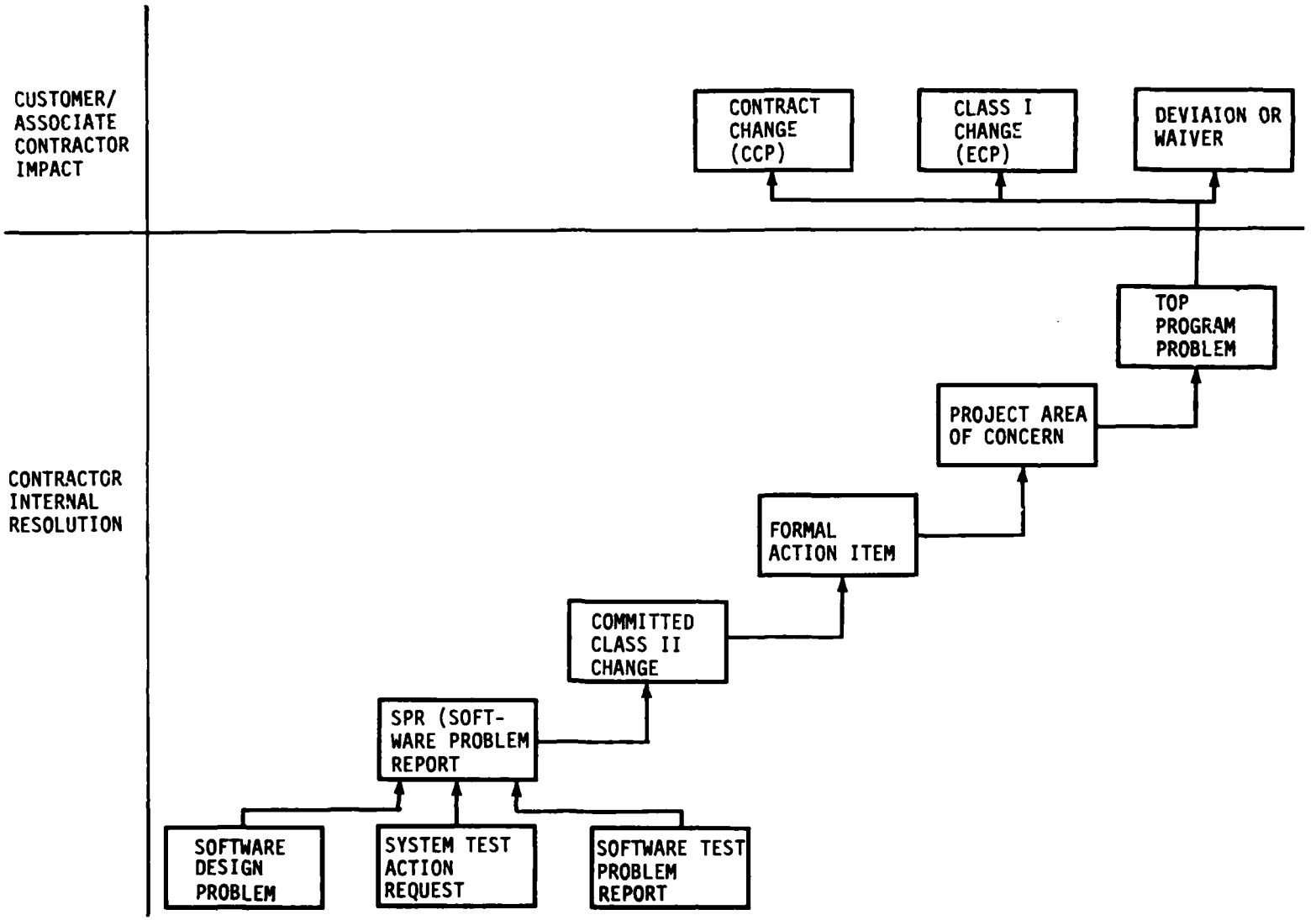


Figure 5.2-1. Problem Abatement Mechanisms

PROGRAM:

Level of problem (Top Program Problem, Area of Concern) NO._____

TITLE:

IDENTIFIED BY AND DATE: Your (identifier) name	**DUE DATE:**	**ACTION MANAGER:** Person to take action for resolution

DESCRIPTION:

(develop action plan, coordinate and implement action to resolution).

Description plus impact if not resolved.

BACKGROUND:

What are requirements, where specified (schedule, SOW, specs, agreements).
Other narrative to fully explain problem.

RECOMMENDED SOLUTION:

What would resolve problem including alternates.

EVIDENCE OF COMPLETION:

or Resolution. Specific statement(s) of criteria that when met will close problem.
(Weekly status and action plans will address this item)

Figure 5.2-2. Area of Concern/Top Program Problem

from developing. Formal Action Items are tracked by the schedule group and require:

a. An assigned action manager

b. Evidence of completion

c. Recommended closure

d. Action plan

e. Due dates

Occasionally major impact problems are encountered which require top management attention:

a. Major change in processor operating system or architecture

b. Insufficient memory

c. Undefined or ambiguous requirements

d. Loss of critical skills

These problems cannot be solved on a simple SPR or Formal Action Item. They are presented to the appropriate levels of contractor management on an "Area of Concern" or "Top Program Problem." The criteria for an "Area of Concern" is that the problem has not yet impacted the program but could if unresolved by a certain date. Areas of concern are presented on view foils to the project manager and his staff and if accepted as legitimate, they are given a control number and tracked to resolution. Resolution involves whatever is necessary to solve the problem. A mini-network may be required and a "back-up" plan together with associated schedules. These "Areas of Concern" are reviewed weekly at schedule reviews, are statused and either continued, suspended, or closed.

Problems which cannot be resolved by any of these means and which will probably impact contractual requirements, costs, or schedule milestones, are elevated to the category of "Top Program Problems." Top program problems are given top management emphasis.

Upon acceptance by the program manager, the schedule maintenance organization is responsible for formal tracking, statusing and graphic displays. These problem abatement mechanisms are treated with the same degree of formality and intensity of attention as Tier I schedules.

In addition to reviewing and statusing known problems, each functional manager is required to canvass his organization at least monthly for candidate "Areas of Concern." If none exists, he is required to sign off to that effect on a monthly "Canvass of Problem of Identification" form. This action forces functional managers to examine their operation and encourages the reporting of deliquencies.

In addition to all of the foregoing, military contractors today are being forced to manage quality costs with greater formality. Quality costs are costs associated with inspection, test, scrap, rework, repetitive failure, warranty claims, etc. MIL-STD-1520A (Corrective Action and Disposition System for Nonconforming Material) is becoming a standard requirement on new contracts. This standard requires the formal documentation of quality costs and the establishment of failure preventive mechanisms such as the "Corrective Action Board." Since software is a deliverable product (DOD 5000.29) it falls within the purview of MIL-STD-1520A. Excessive failures in any given category (i.e. logic, data base, computation, I/O) deserve attention for corrective action. Problem trend analyses and reports designed for this purpose are an integral part of measuring and reporting software status.

Section 6.0 ATE & TS MEASUREMENT AND REPORTING VARIANTS

There are numerous variants in the development of ATE and TS software which should be considered in the early stages of system acquisition planning. Consideration of these variants can help USAF and contractor management plan early warning systems for avoiding pitfalls which can seriously impact the project, if not detected and remedied early. This section describes some of these variants which should be considered during ATE and TS software schedule and performance status reviews.

6.1 ATE VARIANTS

Some of the major ATE unique considerations which should be examined and planned for early in system development are described below.

6.1.1 Integration of Software Components

Component-wise ATE software is a composite of vendor and contractor developed programs. The control and support programs are usually developed by the vendor well in advance of the UUT programs. When finally tested as an integrated entity, incompatibilities are frequently encountered requiring numerous changes. Early integration of UUT programs with vendor software help detect major incompatibilities before they cause insignificant impact.

6.1.2 Change Management.

Change management is generally more difficult for ATE programs than for TS programs simply because of the higher volume and larger number of sources of change. Nearly every UUT design change will precipitate an ATE program change, whereas only significant performance changes necessitate TS program changes. Moreover numerous UUT design groups both within the contractor's organization and his subcontractors contribute to UUT program designs and change. This aggravates the software configuration control problem.

For example, because of budget pressures, the Department of Defense, Joint Services, and industry are encouraging commonality among deployed ground systems users. In the ATE world, configuration commonality is a means to an end; namely lower life cycle costs. The value of equipment commonality is significantly evident when examining the costs incurred in acquiring and maintaining intermediate level maintenance capability for a large number of bases and installations. UUTs developed by literally hundreds of subcontractors must be diagnosed, fault isolated, repaired and retested within minimal turnaround times. The cost of developing, acquiring and maintaining hundreds of sets of UUT peculiar ATE systems would be staggering. To reduce these procurement costs, ATE commonality must be attained while maximizing the universality (capability to test large numbers of different UUTs) of the ATE. Commonality serves the following objectives:

a. Minimization of development costs

b. Minimization of procurement costs

c. Transportability of like configurations among installations

d. Minimal maintenance costs

Preserving this commonality, however, becomes a monumental "headache" if users and UUT subcontractors are continuously changing requirements.

For every UUT change necessitating an ATE change, all like configurations deployed at all sites must be retrofitted to preserve this commonality. Attempts to increase the "universality" of the tester complicates preserving commonality because all using sites must have identical configurations even though a given site might not need that

capability. This creates horrendous configuration management problems, especially if serious design deficiencies result in significant post- deployment changes.

6.1.3 Design for Testability

Management visibility systems must assure that design groups and subcontractors "design and testability," maximizing use of Built-In-Test-Equipment (BITE) to ease the UUT program development task. Designing for testability is usually ignored in the early stages of prime system acquisition. This aspect of UUT development, usually addressed in procurement specs as a design goal, is, in general, not statused as a measurable design parameter. Self-test subroutines designed into the UUT micro-code can immensely simplify the UUT programming task downstream while simultaneously enhancing fleet maintenance.

6.1.4 ATE Growth Potential

As UUT functional capability expands in terms of more functions per UUT or more UUTs, ATE software capabilities must be similarly expanded. For example, extending the range of an air vehicle could involve additional modes of navigation, more complex operational algorithms, faster CPU's, and, perhaps additional defensive avionics capability. Unless this potential growth is anticipated and planned for early in the ATE system acquisition cycle, the required growth potential may not be provided. Memory and timing utilization charts such as Figure 5.1-1 include, therefore, a "design threshold" which should not be exceeded to allow for anticipated growth.

6.1.5 Proprietary Software

While both ATE and TS systems will generally embody some proprietary software (part of the vendor's operating system), its impact on functionality and maintenance of the end item CPCI is potentially greater for ATE than for TS. This is because many ATE operating systems today embody stimuli/measurement routines within the vendor supplied operating system itself. The contractor developed UUT program in ATLAS merely calls up these vendor routines assuming inputs are validly calibrated and applied and that outputs are validly measured. A greater percentage of the actual test process is executed by vendor developed code than by contractor developed code. In TS software the operational program performance is more dependent on contractor developed code than on vendor code. Proprietary software, therefore, is of special importance to ATE software development and hence should be reported and considered as a potential "risk" area because of the difficulty in getting documentation necessary for organic maintenance capability.

6.2 TS VARIANTS

In addition to the previously discussed differences in structure, design, and manageability of TS from that of ATE software, the following considerations uniquely apply to TS.

6.2.1 Modularity of Design

Trainer software, because of its size and complexity (unlike a given UUT program) is not a one man effort. TS software development must be developed in many cases, concurrent with hardware. This among other reasons, necessitates modularity of design so that the many designs can be accomplished on schedule. This demands well structured and documented inter-element interfaces. The principles of top down structured programming characterized by one-input, one-output control structures greatly aid in software manageability. Such techniques allow so called "plug-in compatible" software designs without introducing new connections with the rest of the design. Such design provides disciplined use of the "GO TO" statement which if uncontrolled can introduce many program looping possibilities which

makes correctness checking and maintenance difficult. Management enforcement of disciplined rules for design modularity and structuredness is a consideration unique to TS and other real time software development and statusing.

6.2.2 HOL vs Assembly Language

The question of HOL vs. assembly language is also of interest in modern simulator systems. ATE programs, conversely, are written exclusively in ATLAS or some offshoot thereof. A HOL has the obvious advantage of maximizing a programmer's productivity by allowing him to generate the maximum number of lines of machine code in a given time. Balanced against this benefit however must be the cost of and time expended by a compiler. Compilers, however, automatically take care of problems like keeping track of multiple nested loop variables which must share the same index register, a problem burdensome to the assembly language programmer. Nevertheless, experienced assembly language programmers may be able to save core space and execution time by, for instance saving the index register's contents in a register not referenced by an inner loop. The resulting program may execute faster and occupy less core than equivalent compiler produced code. Studies have confirmed the following:

a. Compiler produced code can be from 1.15 to 3 times as voluminous as an equivalent assembly language program.

b. Execution time for compiler produced code can be from 1.15 to 4 times longer than an assembly language program for the same algorithm.

Therefore, if the option is available, TS software project management needs to insure that trades are conducted early in the detail design phase - whether to write smaller and faster programs in assembly language while incurring a greater programming cost or whether to employ HOL sacrificing execution time, core space and compiler expense. Usually the answer to this trade is "quantity of systems" dependent. Assembly language tends to be more economical as the number of systems deployed increases. If, for example, using an interpretive compiler requires an extra memory chip that costs $20.00 and 100 systems are to be built, then the contractor can spend an additional $2000 to produce the same program in assembly language not even considering the saving in execution time.

Unlike ATE, therefore, management trades such as this must be made early to insure a viable and economical approach to TS software design is undertaken.

6.2.3 Incompatibility of TS Components

A complete system simulator is composed of a basic computer central core from Vendor A, peripherals from Vendor B and operational programs written by the contractor. Incompatibilities between the CPU and, for example, a terminal are not uncommon. It is the responsibility of the prime contractor to integrate all TS components and resolve these problems early. Again, early warning reporting systems are vital to this effort. It is not meant to imply that ATE systems are free of component incompatibility problems; however, for a TS system there is a greater technical challenge for the contractor as a system integrator primarily because of the greater contractor role as the operational software developer. Hence management visibility systems should mandate early system compatibility tests and appropriate reporting mechanisms.

6.2.4 Non-Standard Design

TS software design is less adaptable to standardization of design than ATE software. For example, the central core of an ATE system can be designed and built independent of the application. A standard stimuli/measurement signal repertoire is available "off-shelf" from the ATE vendor. The ATE central core composed of CPU, memory Input/Output (I/O), stimulus generators/measurement devices provide and measure AC, DC, pulse generation, wave form synthesis

over a somewhat standard range of amplitudes and frequencies, irrespective of the application. TS systems, on the other hand are designed around a given training mission where the unique software portion is a greater proportion of the total trainer software package than it is for an ATE package. Accordingly, more management emphasis must be placed on developmental testing in anticipation of a higher risk factor due to the unknowns associated with non-standard and unproven software systems.

Section 7.0 BIBLIOGRAPHY

The following documents are applicable to the subjects covered by this guidebook.

MIL-STD 483, Configuration Management Practices for Systems Equipment, Munitions and Computer Programs

MIL-STD 1521A, Technical Reviews and Audits for Systems, Equipment and Computer Programs

AFR 800-2, Program Management

Embedded Computer Resources and the DSARC Process, Office of the Secretary of Defense, 1977 Edition

Boeing Computer Services Final Technical Report, dated 23 Feb 77, Prepared under contract F30602-76-C-0174 for the Air Force Rome Air Development Center

AFR 800-14 Management of Computer Resources in Systems

SSD Exhibit 61-47B - Computer Program Subsystem Development Milestones

Mini-Micro Systems, Aug 77, Tech Profile, Assembler, Compiler or Interpreter, by Carol A. Ogdin

Section 8.0 MATRIX - GUIDEBOOK TOPICS VS GOVERNMENT DOCUMENTS

The elements of Figure 8.0-1 correspond to sections in the guidebook wherein the corresponding topic is primarily discussed.

TOPICS \ GOVERNMENT PUBLICATIONS	MIL-STD-483	MIL-STD-1521A	AFR 800-2	AFR 800-14	SSD EXHIBIT 61-47B
PROGRAM MILESTONES					3.1 3.2
TECHNICAL REVIEWS AND AUDITS	4.1 4.2	4.2			
MANAGEMENT REPORTING			4.2	4.2	
SCHEDULING			3.3	3.3	
STATUS MEASUREMENT PARAMETERS	4.1				

Figure 8.0-1. Guidebook Topics Versus Government Documentation

Section 9.0 GLOSSARY OF TERMS

Algorithm - A set of rules or processes for solving a problem in a finite number of steps. This software procedure can be presented to a computer precisely and in a standard form, the computer then taking the algorithm's course of action to solve the problem.

Computer Program - A series of instructions or statements in a form acceptable to computer equipment, designed to cause the execution of an operation or series of operations. Computer programs include such items as operating systems, utility programs, and maintenance/diagnostic programs. They also include applications programs such as payroll, inventory, control, operational flight, strategic, tactical, automatic test, crew simulator and engineering analysis programs. Computer programs may be either machine dependent or machine independent, and may be general purpose in nature or be designed to satisfy the requirements of a specialized process of a particular user.

Computer Program Development Cycle - The computer program development cycle consists of six phases: analysis, design, coding and checkout, test and integration, installation and operation and support. The cycle may span more than one system acquisition life cycle phase or may be contained in any one phase. (AFR 800-14, Volume II)

Contract - A legally enforceable agreement between two parties (AF/Contractor, Contractor/subcontractor) which describes a program for product acquisition. The contract contains the System Specifications, the Statement of Work, (SOW), the Contract Data Requirements List (CDRL), and the Work Breakdown Structure (WBS).

Computer Program Component - A subset of a computer program configuration item that takes the form of a module subroutine, program unit, etc.

Computer Program Configuration Items - A computer program or aggregate of related computer programs designated for configuration management. A CPCI may be a punched deck of cards, paper or magnetic tape or other media containing a sequence of instructions and data in a form suitable for insertion in a digital computer.

Configuration Item - An aggregation which satisfies an end use function and is designated for configuration management.

Configuration Management - A management discipline applying technical and administrative direction and surveillance to:

(a) Identify and document the functional and physical characteristics of a configuration item;

(b) Control changes to those characteristics; and

(c) Record and report change processing and implementation status.

Control Software - Software used during execution of a test program which controls the nontesting operations of the ATE. This software is used to execute a test procedure but does not contain any of the stimuli or measurement parameters used in testing a unit under test. Where test software and control software are combined in one inseparable program, that program will be treated as test software. (AFLC 66-37)

Data Base - A collection of program code, tables, constants, interface elements and other data essential to the operation of a computer program or software subsystem.

Estimating Model - A graphical or mathematical representation (of a specific work task) which is utilized to calculate the approximate cost to develop and/or produce a desired product.

Life Cycle Analysis - An analysis of a systems total cost to the government over its full life. It would include the cost of development, production, operation, support, and if applicable, disposal.

Logic Flow - A diagrammatic representation of the logic sequence for a computer program. Logic flows may take the form of the traditional flow charts or in some other form such as a program design language.

Milestone - An event which occurs at a defined time point in a schedule of planned activities.

Module - A portion of the entire software system that normally is one or more programs to accomplish a specified task.

Organic - A term used to designate a task performed by the Air Force rather than a contractor.

Program Design Language - An English-like, specially formatted, textual language describing the control structure, logic structure, and general organization of a computer program. Essential features of a program design language are:

(a) It is an English-like representation of a computer procedure that is easy to read and comprehend.

(b) It is structured in the sense that it utilizes the structured programming control structures and indentation to show nested logic.

(c) It uses full words or phrases rather than the graphic symbols used in flow charts and decision tables.

Program Planning and Control (PP&C) - A contractor and/or SPO organization whose responsibility includes development and maintenance of schedules and other management visibility and control information.

Quality Assurance - A planned and systematic pattern of all software related actions necessary to provide adequate confidence that computer program configuration items or products conform to established software technical requirements and that they achieve satisfactory performance.

Software - A combination of associated computer programs and computer data required to enable the computer equipment to perform computational or control functions.

Source Selection - The process of selecting which among competing contractors shall be awarded a contract. A significant portion of this involves evaluation of proposals to determine the degree to which the government's requirements would be satisfied.

Support Software - Auxiliary software used to aid in preparing, analyzing and maintaining other software. Support software is never used during the execution of a test program on a tester, although it may be resident either on-line or off-line. Included are assemblies, compilers, translators, loaders, design aids, test aids, etc. (AFLC 66-37)

System Engineering - The application of scientific and engineering efforts to transform an operational need or statement of deficiency into a description of systems requirements and a preferred system configuration that has been optimized from a life cycle viewpoint. The process has three principal elements: functional analysis, synthesis, and trade studies or cost-effectiveness optimization.

Test Software - Programs which implement documented test requirements. There is a separate test program written for each distinct configuration of unit under test (AFLC 66-37).

Validation - Computer program validation is the test and evaluation of the complete computer program aimed at ensuring compliance with the performance and design criteria.

Verification - Computer program verification is the iterative process of continuously determining whether the product of each step of the computer program acquisition process fulfills all requirements levied by the previous step, including those set for quality.

Unit Development Folder - A storage folder maintained by a computer programmer where essential listings, schedules, documentation and data are maintained for purposes of visibility and management approval.

Work Breakdown Structure - A standard method for structuring a program into its various required work tasks. A Work Breakdown Structure is implemented per MIL-STD-881A under the guidance in AFR 800-17. When subdivided as necessary by the contractor to identify tasks associated with a single responsible organization, it provides a basis for contract planning, status determination, and reporting.

Section 10.0 ABBREVIATIONS AND ACRONYMS

ACWP	Actual Cost of Work Performance
AFAL	Air Force Avionics Lab
AFLCP	Air Force Logistics Command Pamphlet
AFPRO	Air Force Plant Representative Office
AFR	Air Force Regulation
AFSCP	Air Force Systems Command Pamphlet
ASPR	Armed Forces Procurement Regulation
ASUPT	Advances Simulator Undergraduate Pilot Training
ATE	Automatic Test Equipment
ATLAS	Abbreviated Test Language for all Systems
ATPG	Automatic Test Program Generation
BCWP	Budgeted Cost of Work Performed
BCWS	Budgeted Cost of Work Scheduled
BITE	Built in Test Equipment
CCP	Contract Change Proposal
CDR	Critical Design Review
CDRL	Contract Data Requirements List
CER	Cost Estimating Relationship
CFSR	Contract Funds Status Report
CGI	Computer Generated Imagery
CI	Configuration Item
CMP	Configuration Management Plan
CPC	Computer Program Component
CPCI	Computer Program Configuration Item
CPDP	Computer Program Development Plan
CPR	Cost Performance Report
CPU	Central Processing Unit
CRISP	Computer Resources Integrated Support Plan
CRM	Contract Responsibility Matrix
CRT	Cathode Ray Tube
C/SCSC	Cost/Schedule Control Systems Criteria
C/SSR	Cost/Schedule Status Report
DBMS	Data Base Management System
D&D	Design and Development
DID	Data Item Description
DOD	Department of Defense
DODI	Department of Defense Instruction
ECD	Estimated Completion Date
ECP	Engineering Change Proposal
ESD	Electronic Systems Division
FACI	First Article Configuration Inspection
FCA	Functional Configuration Audit
FQR	Formal Qualification Review
FQT	Formal Qualification Test
FY	Fiscal Year

GRC	General Research Corporation
HEW	Health, Education and Welfare
HOL	Higher Order Language
IBM	International Business Machine Company
ICD	Interface Control Drawings
IMS	Integrated Management System
I/O	Input/Output
ITA	Interface Test Adapter
LOE	Level of Effort
LRU	Line Replaceable Unit
LSI	Large Scale Integrated Circuitry
MEAC	Management Estimate at Completion
MIC	Management Information Center
NAA	North American Autonetics
O&M	Operational and Maintenance
O&S	Operational and Support
PAR	Problem Analysis Report
PCA	Physical Configuration Audit
PDR	Preliminary Design Review
PIDS	Prime Item Development Specifications
PMP	Program Management Plan
PMR	Program Management Review
PQT	Preliminary Qualification Test
PP&C	Program Planning and Control
PROM	Programmable Read Only Memory
R&DA	Requirements Definition and Analysis
R&M	Reliability/Maintainability
RCA	Radio Corporation of America
RDT&E	Research, Development, Test and Evaluation
RF	Radio Frequency
RFP	Request for Proposal
ROC	Required Operational Capability
ROM	Read Only Memory
RSS	Regulations, Specifications and Standards
SAE	Software Acquisition Engineering
SDC	System Development Corporation
SIRD	Software Implementation Requirements Document
SOW	Statement of Work
SPO	Systems Program Office
SDR	System Design Review
SPR	Software Problem Report
SRR	System Requirements Review
STOL	Short Take Off and Landing
TI	Technical Interchange
T.O.	Technical Order
TRD	Test Requirement Document
TS	Training Simulator
UDF	Unit Development Folder
UUT	Unit Under Test

VAR	Variance Analysis Report	V&V	Validation and Verification
VDD	Version Description Document	VV&C	Verification, Validation and Certification
VECP	Value Engineering Change Proposal	WBS	Work Breakdown Structure

Section 11.0 SUBJECT INDEX

*U.S.Government Printing Office: 1980 — 657-084/531